Bayesian Methods for Discovering Structure in Neural Spike Trains

Bayesian Methods for Discovering Structure in Neural Spike Trains

A DISSERTATION PRESENTED
BY
SCOTT WARREN LINDERMAN
TO
THE JOHN A. PAULSON SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2016

Thesis advisors: Professors Ryan P. Adams and Leslie G. Valiant Scott Warren Linderman

Bayesian Methods for Discovering Structure in Neural Spike Trains

Abstract

Neuroscience is entering an exciting new age. Modern recording technologies enable simultaneous measurements of thousands of neurons in organisms performing complex behaviors. Such recordings offer an unprecedented opportunity to glean insight into the mechanistic underpinnings of intelligence, but they also present an extraordinary statistical and computational challenge: how do we make sense of these large scale recordings? This thesis develops a suite of tools that instantiate hypotheses about neural computation in the form of probabilistic models and a corresponding set of Bayesian inference algorithms that efficiently fit these models to neural spike trains. From the posterior distribution of model parameters and variables, we seek to advance our understanding of how the brain works.

Concretely, the challenge is to hypothesize latent structure in neural populations, encode that structure in a probabilistic model, and efficiently fit the model to neural spike trains. To surmount this challenge, we introduce a collection of structural motifs, the design patterns from which we construct interpretable models. In particular, we focus on random network models, which provide an intuitive bridge between latent types and features of neurons and the temporal dynamics of neural populations. In order to reconcile these models with the discrete nature of spike trains, we build on the Hawkes process — a multivariate generalization of the Poisson process — and its discrete time analogue, the linear autoregressive Poisson model. By leveraging the linear nature of these models and the Poisson superposition principle, we derive elegant auxiliary variable formulations and efficient inference algorithms. We then generalize these to nonlinear and nonstationary models of neural spike trains and take advantage of the Pólya-gamma augmentation to develop novel Markov chain Monte Carlo (MCMC) inference algorithms. In a variety of real neural recordings, we show how our methods reveal interpretable structure underlying neural spike trains.

In the latter chapters, we shift our focus from autoregressive models to latent state space models of neural activity. We perform an empirical study of Bayesian nonparametric methods for hidden Markov models of neural spike trains. Then, we develop an MCMC algorithm for switching linear

dynamical systems with discrete observations and a novel algorithm for sampling Pólya-gamma random variables that enables efficient annealed importance sampling for model comparison.

Finally, we consider the "Bayesian brain" hypothesis — the hypothesis that neural circuits are themselves performing Bayesian inference. We show how one particular implementation of this hypothesis implies autoregressive dynamics of the form studied in earlier chapters, thereby providing a theoretical interpretation of our probabilistic models. This closes the loop, connecting top-down theory with bottom-up inferences, and suggests a path toward translating large scale recording capabilities into new insights about neural computation.

Contents

Listing of figures

Acknowledgments

I am extraordinarily fortunate to have had two fantastic advisors, Leslie Valiant and Ryan Adams. Les has been an amazing source of wisdom, perspective, and inspiration. I am thankful for his patience and his guidance in sifting through the excitement of the moment and identifying lasting questions. The first class I attended at Harvard was Ryan's first lecture as a professor, and that fortuitous meeting kicked off five of the most rewarding years of my life. Of all that he has shared with me — a wealth of technical knowledge, an ability to articulate ideas, and an appreciation for clear figures and neat handwriting — I am most grateful for the joy he has helped me find in research.

I also wish to thank Jonathan Pillow, Sam Gershman, and Haim Sompolinsky, who have been invaluable sources of mentorship and advice. Their generous sharing of time and ideas has had a profound impact on my perspective of computational neuroscience.

I cannot imagine a more fun and intellectually stimulating community than the Harvard Intelligent Probabilistic Systems group. I am grateful to David Duvenaud, Miguel Hernandez-Lobato, Dougal Maclaurin, Andy Miller, Oren Rippel, Yakir Reshef, and Jasper Snoek for all the "unsupervised learning" I've benefitted from in MD209. I'd particularly like to thank Matt Johnson, who has literally taught me everything I know about many subjects. His enthusiasm and encouragement have made me a better researcher and a stronger person.

I am indebted to Chris Stock and Aaron Tucker for working with me during their undergraduate years. Whatever I may have taught them about computational neuroscience surely pales in comparison to what they taught me about advising. Their hard work has shaped many pages of this thesis.

I thank my friends in the theory group — Varun, Thomas, Mark, Jon, and Justin — for pushing me to think abstractly, and I thank my housemates in Cambridge — Ryan, Bill, James, Chris, Eddie, and Steve — for keeping me grounded. My graduate years have been blessed by many other wonderful friends, and I thank them for all of the fond memories.

My parents, Richard and Linnea, have always encouraged me to pursue this dream, and I am forever grateful for all the opportunities they have given me. I thank my brothers, Stephen, Matthew, and Randy, for never letting me miss a beat. To Stephen, who once warned me that if I delayed on graduate school I might not finish before the age of thirty, I say: five months to spare!

Above all, my thanks go to my wife, Anne, who has been my partner throughout this great adventure. This thesis is a testament to her unbounded love and support.

1 Introduction

Neuroscience is undergoing a data-driven revolution. In laboratories around the world, optical tools are illuminating thousands of neurons at a time (Kerr and Denk, 2008). For some organisms, we can now monitor large fractions of the neurons in the brain (Ahrens et al., 2013; Prevedel et al., 2014; Lemon et al., 2015). Not long ago, this immense recording capability would have been mind-boggling to the average practitioner poking around in the dark, hoping to record from a handful of neurons. Complementing these recording capabilities, recent advances in connectomics — the mapping of synaptic connectivity (Sporns et al., 2005) — are providing a more complete atlas of the wiring diagram that underlies neural activity (Lichtman et al., 2008; Helmstaedter et al., 2013; Oh et al., 2014). Likewise, sophisticated monitoring and processing methods (e.g. Berman et al., 2014; Wiltschko et al., 2015) are providing rich measurements of the natural behavior that this activity gives rise to.

These unprecedented technological capabilities are leading to a fundamental paradigm shift. The scientific process of proposing hypotheses, testing them against experimental observations, and revising them accordingly, is becoming increasingly data-driven. Rather than collecting data to test a specific hypothesis, we can now collect large-scale recordings in relatively unstructured experimental setups (e.g. from freely behaving animals) and use statistical structure in the data to guide us toward hypotheses. Of course, there is no "free lunch" — we must still specify what types of structure to look for. For example, when we cluster neuronal data we are hypothesizing that there are different

types of neurons, and when we apply principal components analysis to neural population recordings we are assuming that neural activity reflects a low-dimensional latent state. As we move toward this more data-driven paradigm, we are presented with a challenge: how do we formulate models that capture our intuitions and hypotheses about neural computation, how do we fit these models to large scale recordings, and how do we revise our models based on what we have learned?

This thesis builds a sequence of modeling and inference tools to address this challenge. We build tools for analyzing *neural spike trains*, which are sequences of discrete events in time, and we use probabilistic models to compose flexible hypotheses that allow us to *discover structure* in the dynamics of spike trains. The workhorse of this process is a set of *Bayesian methods*, statistical inference algorithms that take in a neural recording and output a distribution over parameters and variables of our model. We begin by describing each of these these in turn.

1.1 Neurons, Spikes, and Computation

The human brain is a three pound ball of densely packed cells. It is soft, with a consistency like that of tofu. Amazingly, from this curd-like mass of cells, our mental faculties arise. About half of the roughly 170 billion cells in our brain are neurons, widely believed to be the fundamental units of computation (Dayan and Abbott, 2001). Neurons are electrically excitable cells that contain an ionic soup of charged atoms like sodium, potassium, and calcium, which together maintain a difference in electrical potential between the inside and the outside of the cell membrane (Kandel et al., 2000).

In most neurons, the cell membrane is littered with voltage-gated ion channels. At rest, these channels are closed, but if the membrane potential is excited above a certain threshold, some channels rapidly start to open, initiating an action potential. As described in the seminal work of Hodgkin and Huxley (1952), positively charged sodium ions are first to rush into the cell, causing a further increase in membrane potential and leading more channels to open. The upswing in membrane potential eventually causes a reversal in cell polarity and the inactivation of sodium channels. At the same time, potassium channels open, allowing an efflux of positively charged potassium ions. Together, these halt the rising membrane potential and drive it back down toward its resting state. This brief action potential, or "spike," takes place in just a few milliseconds.

The first challenge of deciphering neural computation is understanding how information is encoded in patterns of coordinated spiking activity across neural populations. To first approximation, spikes are discrete events in time. That is, in a short enough window of time, a neuron either does or does not fire an action potential. Much progress has been made in understanding how sensory

neurons digitize continuous signals and how the activity of motor neurons innervates muscles and drives behavior (Rieke et al., 1999), yet how internal states, plans, goals, decisions, and thoughts are encoded remains largely a mystery.

The second challenge is understanding how this information is transformed over time. The neurons in our brain are connected by about 10^{14} synapses (Kandel et al., 2000). When a pre-synaptic neuron fires an action potential, neurotransmitters are released that then bind with receptors on the post-synaptic neuron, causing its ion channels to open and allowing current to flow into or out of the post-synaptic cell. These currents induce brief changes in the membrane potential of the downstream cell called post-synaptic potentials (PSPs). Depending on the direction of current, the PSP will be either excitatory, making the downstream neuron more likely to spike, or inhibitory, suppressing post-synaptic spiking. The probability of neurotransmitter release and the number of post-synaptic receptors together determine the "strength" of the synapse, which is manifest in the amplitude of the PSP (Cowan et al., 2003).

This web of synaptic connections imbues neural circuits with complex dynamics that govern how patterns of neural activity evolve over time, along with the information they encode. It is these dynamics that actually perform computation — the transformation of an input into an output. While these dynamics have been very well studied at the level of single connections between pairs of neurons, the dynamics of large networks of interconnected neurons is at the frontier of research.

Perhaps the most fundamental aspect of intelligence is our ability to learn, to store memories, and to generalize from past experience. In neural circuits, learning is most directly associated with the process of synaptic plasticity. In response to coordinated patterns of pre- and post-synaptic spiking, synapses adapt their strength. This synaptic plasticity leads to changes in the dynamics of neural circuits. In some cases, this plasticity leads to the creation of new associations that support generalization from noisy or partial sensory input. Thus, a third challenge of deciphering neural computation is understanding the processes of learning that cause neural dynamics to evolve in an activity-dependent manner.

Our goal, as neuroscientists, is to address these three challenges — encoding, dynamics, and learning — by searching for patterns in large-scale recordings of neural spike trains, simultaneously recorded time series of spiking activity in populations of neurons. While this type of data is very hard to collect in humans, it is widely collected in worms, fish, flies, mice, rats, monkeys, and a host of other model organisms. Still, these model organisms have taught us a great deal about the workings of human brains and intelligence. This thesis does not focus on the unique aspects of any single organism or brain region, but rather on the general challenges associated with modeling structure in

time series of discrete events that are common to all neural spike trains.

1.2 Discovering Structure: Types, Features, Networks, and States

Discovering structure is primarily about finding meaningful abstractions. While neurons, spikes, and synapses are the elementary building blocks of many models of neural computation, our goal is to connect this level of detail to more abstract theories of computation. Just as our knowledge of *in silico* computation is partitioned into a hierarchy of concepts — from transistors, logic gates, pipelines and processors, to assembly code, operating systems, algorithms and programs — our knowledge of neural computation must also include a hierarchy of concepts and descriptions. Marr (1982) proposed three levels of abstraction: the computational level, which concerns the inputs and outputs of a system; the algorithmic level, which specifies the transformations between inputs and outputs; and the implementation level, which focuses on how these transformations are realized in neural substrates. From this perspective, our goal is to interpolate between neural spike trains, typically associated with implementation level descriptions, and higher level abstractions, like those hypothesized by algorithmic and computational theories. To do so, we compose our spike train models out of a set of simple motifs that appear over and over again in models of neural computation.

1.2.1 Discrete Types and Low-Dimensional Features

Many successes of neuroscience have come from careful cataloging of neural types and features. From the earliest investigations of Ramón y Cajal, it has been clear that the brain is made up of anatomically distinct cell types (Cajal, 1899), but further classifications have been made on the basis of functional distinctions as well. Kuffler (1953) identified the response properties of "on" and "off" ganglion cells in the retina, characterizing the ways in which these cells respond to light. Kuffler's students, Hubel and Wiesel, carried on this work, characterizing simple and complex cells in visual cortex (Hubel and Wiesel, 1962) and eventually winning the Nobel Prize for their discoveries. These are pioneering examples of a common theme in neuroscience: clustering cells into discrete types that facilitate our understanding of neural computation. Indeed, the clustering of cells in the early visual pathway continues to this day (Macosko et al., 2015; Sanes and Masland, 2015).

Complementing these classic examples of discrete subtypes of neurons are similarly compelling examples of neurons characterized by continuous features. Most prominent among these are the place cells of the hippocampus (O'Keefe and Nadel, 1978). These cells fire selectively when an ani-

mal is in a particular location in its environment. Thus, each cell is naturally associated with a continuous position in space. As we seek to make sense of spike trains from other complex neural circuits, discrete latent types and continuous latent features form one of the core building blocks in our probabilistic modeling toolkit.

1.2.2 Networks and Dynamics

Networks play a central role in modern neuroscience: they enable us to reason about complex systems in terms of relationships among their constituent parts. Whether the nodes of the network represent individual neurons in a "connectome" (e.g. Sporns et al., 2005), populations of cells in a neural circuit (e.g. Felleman and Van Essen, 1991), voxels in an fMRI recording (e.g. Friston, 1994), or idealized neurons in a theoretical model (e.g. Hopfield, 1982), networks tell us about the pairwise relationships in large populations of nodes. Once we have extracted such a network, a great deal of intuition can be gleaned from its aggregate properties (Bullmore and Sporns, 2009; Newman, 2003). Moreover, the network itself can often be summarized in terms of latent types and features of nodes that govern how likely any pair of nodes is to be connected (Goldenberg et al., 2010).

We often use networks to represent the dynamics of systems. For example, an edge in the network may represent the influence that activity on one node exerts on the subsequent activity of another. In this way, networks provide a simple summary of complex dynamics. Much of this thesis is devoted to learning network representations from neural spike trains, building on a rich body of existing work on generalized linear models for neural spike trains (Paninski, 2004; Truccolo et al., 2005; Pillow et al., 2008). In these models, nodes correspond to the individual neurons in our dataset, and the edges represent a probabilistic relationship between the spiking activity of one neuron and the future firing rate of its downstream neighbors. By combining latent variable models for the network with dynamics models for its edges, this thesis will construct sophisticated, hierarchical models for dynamical spike trains.

1.2.3 Latent States of Neural Populations

Networks model dynamic data in terms of relationships between nodes, which are typically associated with individual neurons in the generalized linear model. In some cases, however, it is more natural to think about neural dynamics in terms of a latent state that evolves over time but cannot be directly observed. For example, population activity might reflect a low-dimensional, continuous latent state with smooth (Yu et al., 2009) or linear (Smith and Brown, 2003; Paninski et al., 2010)

dynamics, or a discrete latent state with Markovian dynamics (e.g. Jones et al., 2007; Latimer et al., 2015). In these models, the firing rates of individual neurons are a function of the underlying latent state.

By combining these simple building blocks — latent types and features, networks, and dynamic latent states — we can express sophisticated hypotheses about the underlying structure in observed neural spike trains. However, in order to determine this structure, we need to instantiate these hypotheses in the form of a model that we can fit and evaluate. Bayesian probabilistic models and inference algorithms provide a principled means of tackling this problem.

1.3 A Bayesian Approach

Structural hypotheses specify how a set of parameters and latent variables give rise to patterns of neural activity. These hypotheses are naturally formalized in terms of probabilistic generative models, which provide an intuitive way of expressing the joint probability of parameters, latent variables, and observed data. Once formalized, we use the tools of Bayesian inference to compute the posterior distribution over parameters and latent variables implied by a given spike train and our prior beliefs, and use this posterior distribution to gain insight into the structure of the data and potential shortcomings of our theories.

Recent decades have witnessed an explosion of interest in Bayesian machine learning methods, and the development of both theoretical and analytical tools have led to the widespread adoption of these techniques (Bishop, 2006; Murphy, 2012). Nevertheless, performing Bayesian inference in sophisticated models will always be a challenge. The art of probabilistic modeling lies in balancing two objectives: designing models that capture the details of the generative processes underlying our data, while simultaneously ensuring that these models admit efficient inference algorithms. To achieve this balance, we often look for model-specific structure that we can exploit during inference. This thesis is largely dedicated to developing algorithms that capitalize on the properties of models in order to efficiently scale to large recordings.

Figure 1.1 outlines the process of model building. We begin by collecting data, in this case, large-scale recordings of neural spike trains, and building a probabilistic model that captures our intuitions and hypotheses about structure in that data. Given these two ingredients, we perform Bayesian inference to compute the posterior distribution over structural variables and parameters. Visualizing, analyzing, and exploring this posterior often lead to new insights and ways to think about generative processes that can explain further structure in the data. We can also criticize the

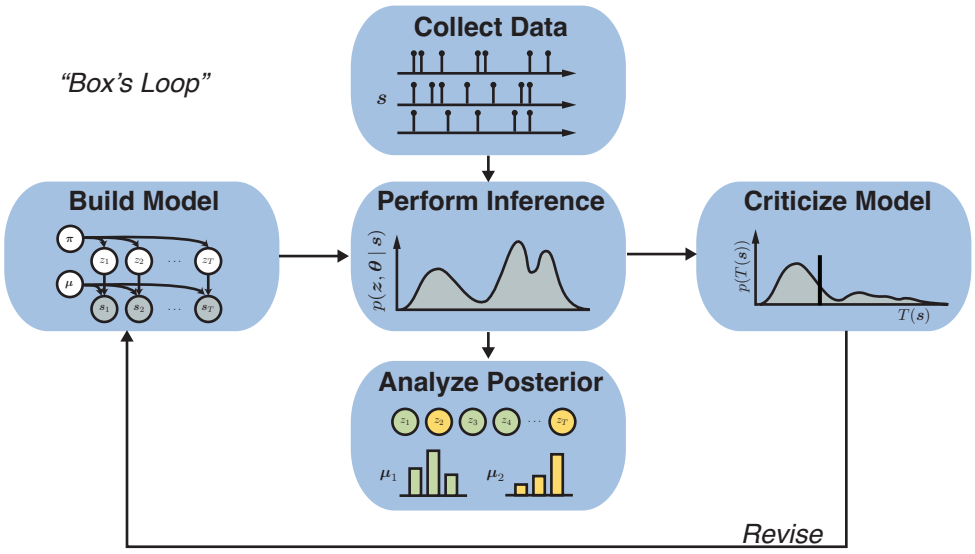

Figure 1.1: Box's loop for hypothesis driven probabilistic modeling. Adapted from Blei (2014).

model, formally, by assessing goodness-of-fit, evaluating predictive likelihoods, and performing posterior predictive checks (Gelman et al., 2013). This leads to new hypotheses, which in turn lead to new models and repetition of this process. Thus, model building is an interactive process in which we are constantly adapting and refining hypotheses. Blei (2014), from whom this figure has been adapted, calls this "Box's loop," a reference to the foundational work of Box (1980).

We are certainly not the first to adopt a Bayesian approach to modeling of neural data. Indeed, probabilistic methods (e.g. Brillinger et al., 1976; Brillinger, 1988; Paninski, 2004; Truccolo et al., 2005; Pillow et al., 2008), particularly Bayesian methods (e.g. Sahani, 1999; Rieke et al., 1999; Yu et al., 2009; Park and Pillow, 2011; Macke et al., 2011), have played a major role in advancing our understanding of spike trains. We build on this foundation and specifically focus on expanding the set of models, combining standard point process and spike count models with hierarchical prior distributions that capture latent structure. To go along with these more sophisticated models, we also derive efficient* Bayesian inference algorithms that take advantage of clever data augmentation strategies. Together, these expand the repertoire of probabilistic models that we may leverage when

*We do not mean efficient in the formal, complexity-theoretic sense. Indeed, approximate Bayesian inference is, in the worst case, NP-hard (Dagum and Luby, 1993; Roth, 1996). We simply mean that the individual iterations of our algorithm can be performed in low-order polynomial time and that these algorithms yield good empirical results in a reasonable, if not provably polynomial, number of iterations.

looking for structure in neural data.

1.4 Summary of Contributions

This thesis consists of a sequence of probabilistic models and inference algorithms that capture increasingly sophisticated structure in neural spike trains. While these models build upon one another, layering in additional structural hypotheses and incorporating more and more detail, the chapters can also be read independently without severe loss of clarity, especially by those who are more versed in machine learning.

We begin with a brief overview of point processes, probabilistic modeling, and Bayesian inference in Chapter 2. This lays the foundation for the methodological contributions in the remainder of this thesis. We also introduce common notation for spike trains, firing rates, and latent variables of the model. Readers who are well versed in probabilistic modeling may skip this chapter without much harm.

Chapter 3 introduces our first probabilistic model for neural spike trains — a combination of continuous time Hawkes processes and probabilistic network models. This is based on Linderman and Adams (2014). However, since this is the first model of the thesis, we provide a more thorough discussion of network models, Hawkes processes, and the steps in deriving an efficient Markov chain Monte Carlo (MCMC) inference algorithm. This is also the only chapter that will apply these techniques to spike trains from domains other than neuroscience, specifically, from finance and criminology.

Chapter 4 continues the focus on Hawkes processes and networks, but here we break from the continuous time models of Chapter 3 and begin working in discrete time. The remainder of the thesis continues in this vein. This chapter also introduces a scalable variational inference algorithm that leverages the discrete model structure. This is based on Linderman and Adams (2015).

Chapter 5 reconsiders the linear interactions of Hawkes processes and develops a nonlinear autoregressive model for neural spike trains. Again, we use probabilistic network models as prior distributions over the pattern of functional interaction, and in doing so, show that interesting representations of neural populations can be learned directly from the data. The key to efficient inference is the Pólya-gamma augmentation — a recently developed method for performing Bayesian inference in models with linear Gaussian structure and discrete observations (Polson et al., 2013). A preliminary version of this work was presented by Linderman et al. (2015).

Chapter 6 continues the network theme, but here we begin to transition from static models to

ones with dynamic latent states. Specifically, we adopt a mechanistic view of the network and treat its connections as actual synapses. This leads to a consideration of the evolution of synaptic weights over time. We construct a framework for modeling arbitrary synaptic plasticity rules and fitting them with particle MCMC. This chapter is based on Linderman et al. (2014).

Chapter 7 departs from network models and instead considers dynamical discrete state space models, specifically hidden Markov models (HMM), for neural data. We address a major challenge in applying these well-known models, namely, selecting the number of latent states. By introducing a hierarchical Dirichlet process (HDP) prior and both an MCMC and a variational inference algorithm, we develop a Bayesian nonparametric solution to this problem. While this combination is relatively well studied, these models are notoriously sensitive to hyperparameter settings. Thus, we introduce a variety of methods for selecting hyperparameters and perform a thorough comparison on real and synthetic data. This is based on Linderman et al. (2016a).

Building on the discrete state space models of the preceding chapter, Chapter 8 develops efficient Bayesian inference algorithms for switching linear dynamical systems (SLDSs) with discrete count observations. The Pólya-gamma augmentation once again makes Bayesian inference surprisingly easy. Given these auxiliary variables, a host of tools for inference in Gaussian models is at our disposal. We then turn to a major problem, that of model comparison. We derive a novel sampling algorithm for the Pólya-gamma distribution that renders annealed importance sampling (AIS) simple and efficient for a broad class of models, including the SLDSs of this chapter. A preliminary version of this work was presented in Linderman et al. (2016b).

Chapter 9 takes a step in a radical direction. We build on the probabilistic models of the preceding chapters, but here we combine these models with a top-down theory of neural computation. Specifically, we consider the repercussions of the "Bayesian brain" hypothesis, namely, that neural circuits are performing approximate Bayesian inference in a probabilistic model of the world. We make predictions about the patterns of neural spike trains that would be expected from such a circuit, and show how the methods of previous chapters can be leveraged to reverse engineer probabilistic models from observations of neural spike trains. This work is necessarily speculative, but we believe it suggests a promising path forward as we seek to reconcile computational and algorithmic theories of neural computation with large scale recordings.

Code for these models and inference algorithms, as well as for many of the figures in this thesis, is available at `https://github.com/slinderman`.

2 Background

This chapter lays the foundation for probabilistic modeling of neural spike trains. We start by introducing the language of generative models, which allow us to formalize, in probabilistic terms, our hypotheses about dynamics and low-dimensional structure. The key ingredients are latent variables that reflect the underlying state of the system and conditional distributions that relate these variables to the observed data. Once we understand the basics of this language, we can begin to articulate hypotheses about dynamical data in the form of generative time series models. Section 2.2 enumerates a few common motifs of time series modeling that will be used throughout this thesis. Finally, given a model and an observed spike train, we can invert the model and reason about the posterior distribution over latent variables using Bayesian inference algorithms such as Markov Chain Monte Carlo and mean field variational inference, which are introduced in Section 2.3. At the end of this chapter, we will have the basic foundation necessary to start looking for structure in neural data. The rest of the thesis will build upon this foundation by developing more sophisticated models and increasingly efficient inference algorithms, and by putting them to use on real neural recordings.

2.1 Generative Probabilistic Models

Generative probabilistic models tell a story of how data comes to be. While this story never captures every physical detail, it serves as an idealized version, capturing the essence of the system. For example, when modeling a neural spike train, we will ignore the states of individual ion channels and the

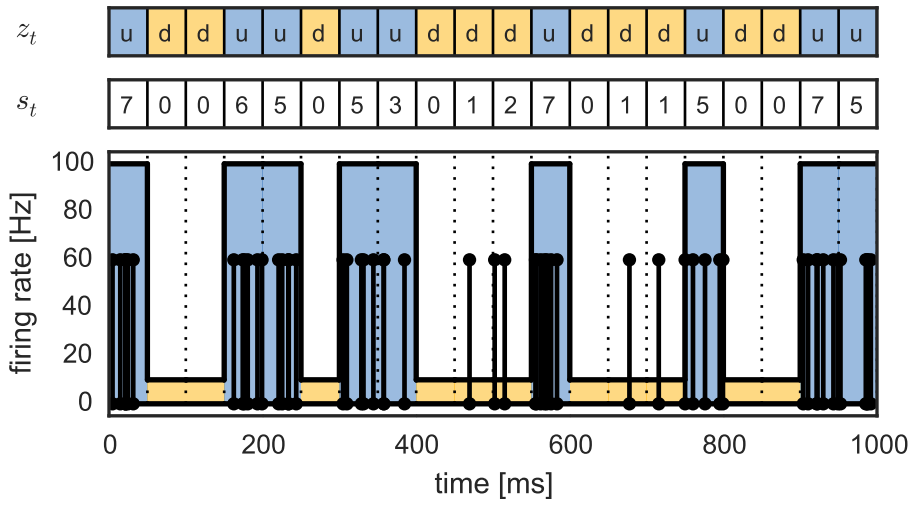

Figure 2.1: A simple neuron that randomly switches between an *up* and a *down* state every 50ms. Here, time bins are colored blue and yellow depending on the latent state, z_t. Each state has an associated firing rate from which a Poisson number of spikes, s_t, is drawn. Precise spike times are uniformly distributed over the 50ms interval.

nonlinear dynamics of membrane potential and instead characterize the instantaneous *firing rate* of a neuron — the probability that a neuron spikes at any moment in time.

As a simple illustration, consider the following generative process. Suppose a neuron has two states, an *up* state and a *down* state. In the *up* state, it spikes at a high rate, say 100Hz, and in the *down* state it fires less frequently, say at 10Hz. Assume that every 50ms the neuron flips a coin to decide its new state and then fires a random number of spikes according to the firing rate associated with that state. For the sake of simplicity, assume the precise spike times are uniformly distributed over the 50ms interval. Once the interval has elapsed, the neuron flips another coin and its rate immediately changes to reflect its new state. Our goal is to infer the latent state of the neuron given the observed spikes.

Clearly, this generative story contains many simplifying assumptions and omits a great amount of detail. In addition to assuming that spiking is adequately captured by firing rates, the notion that a neuron has only two firing rates and that it randomly switches between them is a gross simplification. Nevertheless, this very simple model captures patterns of spiking that have been observed in actual experiments (Cowan and Wilson, 1994; Shu et al., 2003).

We can formalize this generative story with a probabilistic model that specifies a distribution over latent states and observed spike counts. Let $s_t \in \mathbb{N}$ denote the number of spikes counted in the t-th time bin, and $z_t \in \{up, down\}$ denote the corresponding state of the neuron. The assump-

tion that states are drawn from a coin flip corresponds to the prior distribution, $z_t \sim \text{Discrete}(\boldsymbol{\pi})$, where $\boldsymbol{\pi} = [\pi_{\textit{up}}, \pi_{\textit{down}}]$ is a nonnegative vector that sums to one and specifies the probability of *up* and *down* states.* Implicitly, we have assumed that $\boldsymbol{\pi} = \left[\frac{1}{2}, \frac{1}{2}\right]$, though this need not be the case. We previously said that the neurons fire a random number of spikes according to their state-dependent firing rate; now we will formalize this by assuming, $s_t \sim \text{Poisson}(\lambda_{z_t} \cdot \Delta t)$, where $\Delta t = 0.05$s, $\lambda_{\textit{up}} = 100$ spikes/s, and $\lambda_{\textit{down}} = 10$ spike/s.

Figure 2.1 shows a neural spike train sampled from this generative model. The time bins are colored blue or yellow depending on whether the neuron is in the *up* or *down* state, respectively. The precise spike times are denoted by black vertical lines with circular endpoints. Above, the vector of observed spike counts, $\boldsymbol{s} = \left[s_1, \ldots, s_T\right]$, and the vector of latent states, $\boldsymbol{z} = \left[z_1, \ldots, z_T\right]$, are shown. We will use this notation throughout the thesis: bold symbols like $\boldsymbol{s}$ will denote arrays of variables; lowercase bold symbols will typically denote vectors.

The generative procedure defines the *likelihood* of any given set of observed spike counts and corresponding latent states. This can be written as a conditional distribution where the state probabilities and firing rates are given. We have,

$$p(\boldsymbol{s}, \boldsymbol{z} \mid \boldsymbol{\pi}, \boldsymbol{\lambda}) = p(\boldsymbol{z} \mid \boldsymbol{\pi})\, p(\boldsymbol{s} \mid \boldsymbol{z}, \boldsymbol{\lambda}) \tag{2.1}$$

$$= \prod_{t=1}^{T} p(z_t \mid \boldsymbol{\pi})\, p(s_t \mid \lambda_{z_t}) \tag{2.2}$$

$$= \prod_{t=1}^{T} \text{Discrete}(z_t \mid \boldsymbol{\pi})\, \text{Poisson}(s_t \mid \lambda_{z_t} \cdot \Delta t). \tag{2.3}$$

Since Δt is a constant, we do not include it as a random variable in the joint distribution or explicitly condition on it.

The probabilistic model specifies the particular factorization of the likelihood implied by the generative story. Eq. 2.1 applies the product rule of probability, and reflects the assumptions that $\boldsymbol{z}$ depends only on $\boldsymbol{\pi}$ and $\boldsymbol{s}$ depends only on $\boldsymbol{z}$ and $\boldsymbol{\lambda}$. In going from (2.1) to (2.2), we have asserted that the latent states z_t and $z_{t'}$ are conditionally independent given $\boldsymbol{\pi}$, and that the spike counts s_t and $s_{t'}$ are conditionally independent given their corresponding latent states and firing rates. This conditional independence assumption, which was implicit in the generative story, becomes explicit

*The notation $z \sim \text{P}(\theta)$ means that the random variable z is sampled from (or distributed according to) the distribution P, which is parameterized by θ. When we write $\text{P}(z \mid \theta)$ we refer to the density (assuming it exists) of P evaluated at z. A list of commonly used distributions and their densities is given in Appendix A.

when we factor the likelihood into a product over time bins. Eq. 2.3 specifies the functional form of the conditional distributions. When we hypothesize relationships between different variables, we are making assertions about the factorization and the form of the likelihood. In Section 2.2, we explore different patterns of conditional dependence that provide the building blocks of models for dynamic data.

So far, we have assumed that the firing rates and state probabilities are known, but in practice this is a bit unreasonable. To complete the probabilistic model, we need to combine the likelihood function with a *prior distribution* that captures our uncertainty about these parameters. For example, a more reasonable hypothesis is that neurons have two firing rates, and while we do not know their exact values, we can specify a distribution over them, $p(\boldsymbol{\lambda})$. Similarly, we may not know the exact probability of each state, $\boldsymbol{\pi}$, but perhaps we can specify a prior, $p(\boldsymbol{\pi})$, that captures our intuition that the states should be equally likely *a priori*. Putting this all together, we can now write down the *joint distribution* of our probabilistic model — the product of the likelihood and the prior distributions:

$$p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\lambda}) = p(\boldsymbol{s}, \boldsymbol{z} \mid \boldsymbol{\pi}, \boldsymbol{\lambda})\, p(\boldsymbol{\pi})\, p(\boldsymbol{\lambda}). \tag{2.4}$$

When constructing a probabilistic model, we express these prior intuitions and simultaneously make inference easier by using *conjugate* prior distributions.

Conjugate Prior Distributions

A conjugate prior ensures that the conditional distribution of a parameter, given the data, will have a tractable form. Specifically, the conditional distribution will have the same form as the prior. For example, take the parameter, λ_{up}. If we look at the likelihood as a function of λ_{up} and ignore terms that do not depend on this parameter, we have,

$$\begin{aligned} p(\boldsymbol{s}, \boldsymbol{z} \mid \boldsymbol{\pi}, \boldsymbol{\lambda}) &\propto \prod_{t=1}^{T} \left[\text{Poisson}(s_t \mid \lambda_{up} \cdot \Delta t)\right]^{\mathbb{I}[z_t = up]} \\ &\propto \prod_{t=1}^{T} \left[\lambda_{up}^{s_t}\, e^{-\lambda_{up} \cdot \Delta t}\right]^{\mathbb{I}[z_t = up]} \\ &= \lambda_{up}^{s_{up}}\, e^{-\lambda_{up} \cdot t_{up}}, \end{aligned}$$

where

$$s_{up} = \sum_{t=1}^{T} s_t \cdot \mathbb{I}[z_t = up],$$
$$t_{up} = \sum_{t=1}^{T} \Delta t \cdot \mathbb{I}[z_t = up],$$

and $\mathbb{I}[x]$ is an indicator function that equals one if x evaluates to true and equals zero otherwise.

Now consider a gamma prior distribution,

$$\begin{aligned} p(\lambda_{up} \mid \alpha, \beta) &= \text{Gamma}(\lambda_{up} \mid \alpha, \beta) \\ &\propto \lambda_{up}^{\alpha-1} \, e^{-\lambda_{up} \cdot \beta}. \end{aligned}$$

The conditional distribution over λ_{up} given the observed spike counts, the latent states, and the prior is proportional to the likelihood times the prior. This simplifies to,

$$\begin{aligned} p(\lambda_{up} \mid \boldsymbol{s}, \boldsymbol{z}, \alpha, \beta) &\propto p(\boldsymbol{s}, \boldsymbol{z} \mid \boldsymbol{\pi}, \boldsymbol{\lambda}) \, p(\lambda_{up} \mid \alpha, \beta) \\ &\propto \lambda_{up}^{s_{up}+\alpha-1} \, e^{-\lambda_{up}(t_{up}+\beta)} \\ &\propto \text{Gamma}(\lambda_{up} \mid s_{up} + \alpha, \, t_{up} + \beta). \end{aligned}$$

Since both the prior and the the conditional distribution over λ_{up} are in the gamma family, we say gamma prior is conjugate with this product-of-Poissons likelihood. Moreover, the parameters of conditional distribution only depend on $\boldsymbol{s}$ and $\boldsymbol{z}$ through simple *sufficient statistics*, s_{up} and t_{up}. A Dirichlet prior distribution on the state probability, $\text{Dir}(\boldsymbol{\pi} \mid \gamma)$, is similarly conjugate with the product of discrete densities in the likelihood that links $\boldsymbol{\pi}$ and $\boldsymbol{z}$. In fact, conjugate priors exist for all likelihoods in the *exponential family*. These ideas are thoroughly discussed in standard Bayesian statistics and machine learning textbooks like Gelman et al. (2013); Murphy (2012).

Latent Variables, Parameters, and Hyperparameters As our models become increasingly complicated, we will often distinguish between the different types of random variables. The states, $\boldsymbol{z}$, are called *local latent variables* because there is one for each data point. The unknown latent state probability and the firing rates, $\{\boldsymbol{\pi}, \boldsymbol{\lambda}\}$, are either called *parameters* or *global latent variables* because their dimension is fixed. The remaining values, $\{\alpha, \beta, \gamma\}$, are called *hyperparameters*. These are constants that we set prior to performing inference. Typically, these can be tuned by cross-

validation, or simply set based on intuition and physical constraints. For conciseness, we will refer to the set of all parameters as $\boldsymbol{\theta}$ and the set of hyperparameters as $\boldsymbol{\eta}$.

2.1.1 Representations of Spike Trains

One of the first decisions we must make is how to represent our data. In this thesis we will focus on modeling spike trains, which are sequences of discrete events in time. These spike trains typically come from spike sorting algorithms applied to extracellular recordings from multi-electrode arrays (Lewicki, 1998) or from deconvolution algorithms applied to optically recorded calcium fluorescence traces (Pnevmatikakis et al., 2016; Vogelstein et al., 2010). Reducing the data to a set of spike times often results in enormous compression. Rather than considering electrode potentials, which may be sampled at upwards of 10kHz, or calcium fluorescence traces, which are highly autocorrelated due to the relatively slow dynamics of calcium concentration in cells, we only consider the times of action potentials.

The most general representation of a spike train is a set of real-valued times for each neuron. In Figure 2.1, this corresponds to the temporal locations of each black spike. When there is more than one neuron, we have a set of *marked* spike times, which we call,

$$\mathcal{S} = \{(s_m, c_m)\}_{m=1}^{M} \subset [0, T] \times \{1, \ldots, N\}.$$

Each member of this set consists of a real-valued spike time s_m in the interval $[0, T]$, and an integer, $c_m \in \{1, \ldots, N\}$, that specifies the index of the cell that generated this spike. M is the total number of spikes on all neurons.

This continuous-time representation is warranted when the temporal resolution of the data is considerably higher than the time-scale of typical action potentials. For example, multi-elecrode arrays typically have sampling intervals of 0.1ms or smaller, whereas the width of action potentials is on the order of 1ms. This allows us to specify the spike time as an effectively real-valued number.

Sets of discrete events like these are typically modeled as realizations of a *marked point process* (Daley and Vere-Jones, 2003). Such a process is defined by its nonnegative firing rates[†], $\{\lambda_n(t \mid \mathcal{H}_t)\}_{n=1}^{N}$, where $\mathcal{H}_t$ captures the history of the process through time t. For example, the history may include the previous spikes, $\mathcal{H}_t = \{(s_m, c_m) : s_m < t\}$, as well as some external covariates. If we consider a small time window, $[t, t + \Delta t)$, and take the limit as Δt approaches zero, $\lambda_n(t \mid \mathcal{H}_t) \cdot \Delta t$ is the expected number of spikes fired by neuron n in the window $[t, t + \Delta t)$.

[†]In the point process literature, these firing rates are called *conditional intensity functions*.

The limiting perspective on the conditional intensity functions suggests an alternative, discrete-time representation. Rather than modeling a set of continuous spike times and conditional firing rates, we may instead represent a spike count matrix, $\boldsymbol{S}$, and the corresponding rate matrix, $\boldsymbol{\Lambda}$, where,

$$\boldsymbol{S} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,N} \\ \vdots & & \vdots \\ s_{T,1} & \cdots & s_{T,N} \end{bmatrix}, \qquad \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_{1,1} & \cdots & \lambda_{1,N} \\ \vdots & & \vdots \\ \lambda_{T,1} & \cdots & \lambda_{T,N} \end{bmatrix}.$$

Here, $s_{t,n} \in \mathbb{N}$ denotes the number of spikes fired in the t-th time bin by the n-th neuron, and $\lambda_{t,n} \in \mathbb{R}_+$ denotes the corresponding firing rate. Sometimes, the effects we are interested in studying occur at relatively slow time scales, so discretizing may provide valuable compression while retaining most of the relevant information. For example, if we are studying neural dynamics on the order of minutes, then simply knowing how many spikes occurred each second may provide most of the relevant information, while precise, millisecond-resolution spike timing may be superfluous.

However, the primary reason to discretize spike times into a matrix of counts is that the statistics and machine learning community has developed a much broader set of models for matrices than for sets of continuous time events. In the next section, we will explore a number of common modeling motifs that can be applied to time series data represented as matrices, and many of the chapters of this thesis will focus on extending these motifs in novel ways.

2.2 Motifs of Time Series Models

The art of probabilistic modeling lies in balancing two conflicting concerns: our model should capture as much of the relevant structure in the data as possible, drawing on our intuition and our existing knowledge of the system, yet at the same time we wish to limit the complexity of the model so that we may perform inference efficiently. One way to balance these goals is to compose our model out of common, well-studied motifs.

Motifs correspond to factorizations of probabilistic models. To visualize these motifs, we represent the probabilistic model in the form of a directed acylcic graph. Each node in the graph corresponds to a random variable, and shaded nodes indicate which variables are observed. The edges represent conditional dependencies. For example, in the mixture model shown in Figure 2.2, the

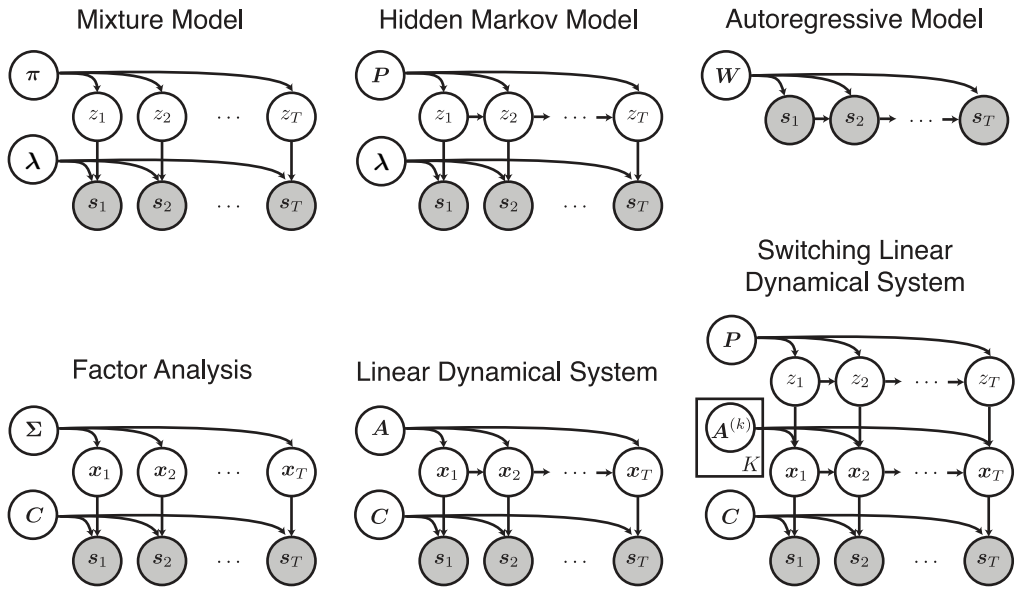

Figure 2.2: Motifs of time series models. By introducing conditional dependencies and layers of random variables, we construct models that reflect sophisticated hypotheses about the structure underlying the data. See Section 2.2 for detailed description.

spike count s_2 has incoming edges from the corresponding latent state z_2 and the firing rates, $\boldsymbol{\lambda}$. Thus, the joint probability distribution contains the factor, $p(s_2 \mid z_2, \boldsymbol{\lambda})$. Since the graph is directed and acyclic, we read off the factors starting with the root nodes, $p(\boldsymbol{\pi})$ and $p(\boldsymbol{\lambda})$, and ending with the leaf nodes, $p(s_t \mid z_t, \boldsymbol{\lambda})$. In this way, the graph captures the factorization of the joint probability distribution and specifies a particular subset of all possible joint distributions over this set of variables.

The edges of the graph do not, however, specify the type of the random variable or functional form of the factors. For example, a node may indicate either a discrete or a continuous random variable, and an edge may indicate an arbitrary form of dependence, like a linear relationship. In this way, two models may share the same graph but have fundamentally different interpretations. This is true of mixture models and factor analysis models shown in Figure 2.2. Some patterns of factorization, types, and dependencies are used over and over again and form the building blocks for more complex models. Next, we discuss a few of the common motifs shown in Figure 2.2.

Mixture Models Our working example from Section 2.1 is an instance of a simple mixture model. The firing rate assumes only two values, and the observed spike counts are a mixture of counts drawn from the *up* state and counts from the *down* state. We can easily extend this to populations of neurons and mixtures of more than two states. Suppose there are now K states, such that $z_t \in \{1, \dots, K\}$. Furthermore, we generalize the rates λ_{up} and λ_{down}, to vectors of rates, one for each neuron and state. In a slight abuse of notation, let $\boldsymbol{\lambda}_k = [\lambda_{k,1}, \dots, \lambda_{k,N}]$ denote a vector of rates in which $\lambda_{k,n}$ is the firing rate of the n-th neuron in state k.

In a mixture model, the latent states are discrete, the time bins are conditionally independent, and the dependence of s_t on z_t and $\boldsymbol{\lambda}$ is linear. To see the latter claim, note that that the instantaneous firing rate of neuron n can be written, $\sum_{k=1}^{K} \mathbb{I}[z_t = k] \cdot \lambda_{k,n}$. These three properties suggest multiple dimensions along which the mixture model may be generalized.

Hidden Markov Models First, let's address the conditional independence of time bins in the mixture model. According to this model, the distribution over latent states factors into a product, $p(\boldsymbol{z} \mid \boldsymbol{\pi}) = \prod_t p(z_t \mid \boldsymbol{\pi})$. This clearly ignores the temporal dynamics of neural data. Instead, we may hypothesis that latent states obey Markovian dynamics,

$$\begin{aligned} p(\boldsymbol{z} \mid \boldsymbol{\pi}^{(0)}, \boldsymbol{P}) &= p(z_1 \mid \boldsymbol{\pi}^{(0)}) \prod_{t=2}^{T} p(z_t \mid z_{t-1}, \boldsymbol{P}) \\ &= \text{Discrete}(z_1 \mid \boldsymbol{\pi}^{(0)}) \prod_{t=2}^{T} \text{Discrete}(z_t \mid \boldsymbol{\pi}^{(z_{t-1})}), \end{aligned}$$

where $\boldsymbol{\pi}^{(0)} \in [0, 1]^K$ is a discrete probability distribution over initial states, and

$$\boldsymbol{P} = \begin{bmatrix} — & \boldsymbol{\pi}^{(1)} & — \\ & \vdots & \\ — & \boldsymbol{\pi}^{(K)} & — \end{bmatrix},$$

is a $K \times K$ transition matrix where the row, $\boldsymbol{\pi}^{(k)} \in [0, 1]^K$, specifies a discrete conditional distribution over z_t given $z_{t-1} = k$. This is known as a hidden Markov model (HMM) (Baum and Petrie, 1966; Rabiner, 1989), and the corresponding graphical model is shown in Figure 2.2. Chapter 7 studies some of the challenges involved in selecting the number of states, K, in a nonparametric way.

Autoregressive Models In an HMM, correlations in spike counts from one bin to the next arise from correlations in the underlying latent states. Alternatively, we may directly model the rate as a function of previous spike counts. For example, consider an autoregressive model with linear dynamics,

$$\lambda_{t,n} = \sum_{n'=1}^{N} \sum_{d=1}^{D} w_{n' \to n}^{(d)} \cdot s_{t-d,n'}. \tag{2.5}$$

The weight, $w_{n' \to n}^{(d)}$, specifies the influence that spikes on neuron n' have on the rate of neuron n at an offset of d time bins in the future. Unlike the HMM, which has an autoregressive model for latent states, here the autoregression governs the rates directly. Moreover, this autoregressive model sums over the spike counts of all neurons over the past D time bins, allowing delayed interactions. Figure 2.2 shows the graph structure of an autoregressive model in the special case that $D = 1$.

In continuous time, autoregressive interactions like these are the basis of the Hawkes process (Hawkes, 1971), a mutually-excitatory point process. Chapter 3 will study these models in great detail, and Chapter 4 will extend the Hawkes process inference algorithms to their discrete time counterparts.

One complication of this formulation is that negative weights could lead to negative firing rates, which would invalidate the assumptions of the model. The easiest way to address this issue is to require nonnegative weights, $w_{n' \to n}^{(d)} \in \mathbb{R}_+$. This implicitly instantiates the hypothesis that interactions between spikes on one neuron and the rate of another is always excitatory — a spike can never decrease the future firing rate. While this is not the most biologically realistic model given our knowledge of excitatory and inhibitory synapses, it is important to remember that this is simply a descriptive model of firing rate dynamics, and it does not necessarily map onto physical synaptic connections. As we will show, the weights inferred by this type of excitatory autoregressive model can still provide useful insight into the structure of neural activity.

Nonlinear Autoregressive Models In order to capture both excitatory and inhibitory autoregressive weights, we need to introduce a nonlinear function that ensures a nonnegative firing

rate. Specifically, assume that,

$$\psi_{t,n} = \sum_{n'=1}^{N} \sum_{d=1}^{D} w_{n' \to n}^{(d)} \cdot s_{t-d,n'},$$
$$\lambda_{t,n} = g\left(\psi_{t,n}\right).$$

The nonlinear function $g(\cdot) : \mathbb{R} \to \mathbb{R}_+$ maps a real valued "activation," $\psi_{t,n}$, into a nonnegative firing rate, $\lambda_{t,n}$. In this formulation, the weights may be either positive or negative to reflect either excitatory or inhibitory interactions, respectively. In computational neuroscience, this is often called a generalized linear model (GLM) (Paninski, 2004; Truccolo et al., 2005; Pillow et al., 2008), since the linear-nonlinear cascade that links spike history to firing rate is an instance of the GLM commonly used in statistics (Nelder and Baker, 1972). Chapter 5 combines these nonlinear autoregressive models with prior distributions on the underlying network and derives efficient Bayesian inference algorithms to fit them to data.

Factor Models HMM's introduced dynamics to the mixture model and nonlinear autoregressive models generalized the linear functional dependence. Factor models generalize the discrete nature of the random variables with a continuous analogue. For example, consider a model in which the discrete variable $z_t \in \{1, \ldots, K\}$ is replaced by a discrete probability distribution $\boldsymbol{\pi}_t \in [0, 1]^K$. The rate is then a nonnegative combination,

$$\lambda_{t,n} = \sum_{k=1}^{K} \pi_{t,k} \cdot \lambda_{k,n}.$$

This is naturally interpreted as a *mixed membership model* in which the rates at each time bin derive from a mixture of discrete latent states with mixing weights $\boldsymbol{\pi}_t$. In text modeling, this motif is the basis of the latent Dirichlet allocation (LDA) model (Blei et al., 2003).

Alternatively, we may replace the discrete latent state with a continuous one, $\boldsymbol{x}_t \in \mathbb{R}^K$. As in the nonlinear autoregressive model, we can retain the linear form and introduce an elementwise

nonlinearity to ensure nonnegative firing rates:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{x}_t \mid \mathbf{0}, \boldsymbol{\Sigma}),$$
$$\psi_{t,n} = \sum_{k=1}^{K} x_{t,k} \cdot c_{k,n},$$
$$\lambda_{t,n} = g(\psi_{t,n}).$$

Here, $c_{k,n}$ is an entry in the real valued matrix $\boldsymbol{C} \in \mathbb{R}^{K \times N}$, and $\boldsymbol{\Sigma} = \operatorname{diag}\left([\sigma_1^2, \ldots, \sigma_K^2]\right)$. This corresponds to a factor analysis model. Unlike standard factor analysis, however, here the observations are discrete spike counts rather than Gaussian observations.

Linear Dynamical Systems In the same way that HMM's extend mixture models with temporal dynamics, linear dynamical systems (LDSs) extend factor models with linear autoregressive dynamics in the latent state. We simply replace the prior on $\boldsymbol{x}$ with a model of the form,

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}_1 \mid \mathbf{0}, \boldsymbol{\Sigma}) \prod_{t=2}^{T} \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{A}\boldsymbol{x}_{t-1}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{A} \in \mathbb{R}^{K \times K}$ specifies the linear dynamics of the latent state. The elementwise nonlinear mapping from latent states to firing rates is the same as in the factor model, but now the linear autoregressive nature of the dynamics induces correlations in spike counts from one time bin to the next.

Hierarchical Extensions These motifs — continuous and discrete latent states, linear autoregressive dynamics, and nonlinear link functions — provide a foundation for constructing probabilistic models for spike trains. Atop this foundation, we may layer additional random variables reflecting hypotheses about shared structure. For example, a switching linear dynamical system, shown in Figure 2.2 and studied in Chapter 8, combines discrete *and* continuous latent states (Murphy, 2012; Fox, 2009). Likewise, Chapters 3, 4, and 5 consider structured prior distributions on the weights of autoregressive models, and Chapter 7 considers nonparametric Bayesian priors on the number of states in an HMM. Once the dynamics model has been specified, it is easy to test a variety of hypotheses about hierarchical structure. In order to fit these models, however, we need efficient inference algorithms that capitalize on the compositional structure of the model.

2.3 Bayesian Inference

Given an observed spike train, our goal is to compute the posterior distribution over latent variables, $\boldsymbol{z}$, and parameters, $\boldsymbol{\theta}$, of the model. For example, in an HMM the latent variables are the dynamic latent states and the parameters are $\boldsymbol{\theta} = \{\boldsymbol{P}, \boldsymbol{\lambda}\}$, the transition matrix and the firing rates for each latent state. Bayes' rule relates the posterior distribution to the joint distribution of our probabilistic model,

$$p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s}) = \frac{p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\theta})}{p(\boldsymbol{s})} = \frac{p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\theta})}{\int p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z} \, \mathrm{d}\boldsymbol{\theta}}. \tag{2.6}$$

Unfortunately, the denominator in Eq. 2.6 involves an integral that is intractable for all but the simplest models. Instead, we must resort to approximate algorithms like Markov chain Monte Carlo (MCMC) and mean field variational inference. We will briefly describe each of these in turn.

2.3.1 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) algorithms are a central component of modern machine learning, and many texts are devoted to the subject (e.g. Geyer, 1992; Gilks, 2005; Robert and Casella, 2013). The fundamental idea is to generate a collection of samples from the posterior distribution and use them to estimate expectations. Specifically, given a set of samples,

$$\left\{ \left(\boldsymbol{z}^{(1)}, \boldsymbol{\theta}^{(1)}\right), \ldots, \left(\boldsymbol{z}^{(L)}, \boldsymbol{\theta}^{(L)}\right) \right\},$$

where

$$\boldsymbol{z}^{(\ell)}, \boldsymbol{\theta}^{(\ell)} \sim p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s}),$$

we can form a Monte Carlo estimate of the expectation of a function $f(\boldsymbol{z}, \boldsymbol{\theta})$ with respect to the posterior,

$$\mathbb{E}_{p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s})} \left[f(\boldsymbol{z}, \boldsymbol{\theta}) \right] \approx \frac{1}{L} \sum_{\ell=1}^{L} f\left(\boldsymbol{z}^{(\ell)}, \boldsymbol{\theta}^{(\ell)}\right).$$

When the samples are independently drawn from the posterior, the strong law of large numbers states that the Monte Carlo estimate converges to the true expectation almost surely, implying that

these Monte Carlo estimates are unbiased. Moreover, if the function f is real-valued, the variance of the Monte Carlo estimator scales as $\mathcal{O}(L^{-1})$ regardless of the dimension of $\boldsymbol{z}$ and $\boldsymbol{\theta}$.

To collect these samples, we design a Markov chain to stochastically explore the space of latent variables and parameters. The chain iteratively samples a new state according to its transition operator, $\mathcal{T}\big((\boldsymbol{z}, \boldsymbol{\theta}) \to (\boldsymbol{z}', \boldsymbol{\theta}')\big)$, which specifies the probability of transitioning from state $(\boldsymbol{z}, \boldsymbol{\theta})$ to state $(\boldsymbol{z}', \boldsymbol{\theta}')$. Each state the Markov chain visits is taken as a sample. If we design the Markov chain appropriately, we guarantee that the transition operator will asymptotically visit states according to their posterior probability.

When states are sampled with a Markov chain, it is no longer true that the samples are independent. In fact, the transition operator often leads to relatively local updates, which in turn lead to autocorrelation in the sequence of samples. This does not affect the bias of the Monte Carlo estimate, but it does affect the constant in the asymptotic $\mathcal{O}(L^{-1})$ convergence rate. However, in addition to the increased variance, MCMC algorithms also suffer from a transient bias due to the fact that the initial state is not drawn from the posterior distribution (we would not be using MCMC if we could sample from the posterior directly). Fortunately, the transient bias of the Monte Carlo estimator also decays as $\mathcal{O}(L^{-1})$. Since the mean squared error of an estimator is equal to its variance plus its bias squared, and since both variance and bias scale inversely with L, the asymptotic effect of the transient bias is insignificant compared to that of the variance.

The critical property of our Markov chain is that, asymptotically, it visits states with probability equal to the true posterior probability. For this asymptotic guarantee to hold, the posterior distribution must be invariant with respect to the transition operator, which is defined by the following equivalence,

$$p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s}) = \int p\big(\boldsymbol{z}', \boldsymbol{\theta}' \mid \boldsymbol{s}\big)\, \mathcal{T}\big((\boldsymbol{z}', \boldsymbol{\theta}') \to (\boldsymbol{z}, \boldsymbol{\theta})\big)\, \mathrm{d}\boldsymbol{z}'\, \mathrm{d}\boldsymbol{\theta}'. \tag{2.7}$$

Intuitively, an invariant, or "stationary" distribution with respect to $\mathcal{T}$ has the property that if we randomly sample a state from the stationary distribution and apply the transition operator, the resulting state will be drawn from the stationary distribution as well. When $\boldsymbol{z}$ and $\boldsymbol{\theta}$ are discrete, the stationary distribution is an eigenvector of a transition matrix with an eigenvalue of one.

In addition to leaving the posterior distribution invariant, the Markov chain must also converge to this stationary distribution regardless of where it starts. If this property holds, the Markov chain is *ergodic*, and the posterior distribution is the unique stationary distribution of the chain. One simple sufficient condition that ensures ergodicity is that the transition probability be strictly positive

for all states.

Designing an MCMC algorithm thus boils down to designing a valid transition operator. This is typically done by composing a sequence of operators, $\mathcal{T} = \mathcal{T}_1 \circ \ldots \circ \mathcal{T}_K$, each of which leaves the stationary distribution intact. While there are many ways of developing these transition operators, one of the most common is to sample from the conditional distribution of one variable while holding the rest fixed. This leads to an algorithm called Gibbs sampling (Geman and Geman, 1984).

Gibbs Sampling

Consider a transition operator, $\mathcal{T}_{\boldsymbol{z}}$, that only updates $\boldsymbol{z}$, holding $\boldsymbol{\theta}$ constant. In order to update $\boldsymbol{z}$, it samples from the conditional distribution, $p(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{s})$. To see that this transition operator leaves the posterior distribution invariant, we plug it into (2.7):

$$\begin{aligned}
&\int \mathcal{T}_{\boldsymbol{z}}\big((\boldsymbol{z}', \boldsymbol{\theta}') \to (\boldsymbol{z}, \boldsymbol{\theta})\big)\, p\big(\boldsymbol{z}', \boldsymbol{\theta}' \mid \boldsymbol{s}\big)\, \mathrm{d}\boldsymbol{z}'\, \mathrm{d}\boldsymbol{\theta}' \\
&\qquad = \int p(\boldsymbol{z} \mid \boldsymbol{\theta}', \boldsymbol{s})\, \delta_{\boldsymbol{\theta}'}(\boldsymbol{\theta})\, p\big(\boldsymbol{z}', \boldsymbol{\theta}' \mid \boldsymbol{s}\big)\, \mathrm{d}\boldsymbol{z}'\, \mathrm{d}\boldsymbol{\theta}' \\
&\qquad = \int p(\boldsymbol{z} \mid \boldsymbol{\theta}', \boldsymbol{s})\, \delta_{\boldsymbol{\theta}'}(\boldsymbol{\theta})\, p\big(\boldsymbol{\theta}' \mid \boldsymbol{s}\big) \underbrace{\int p\big(\boldsymbol{z}' \mid \boldsymbol{\theta}', \boldsymbol{s}\big)\, \mathrm{d}\boldsymbol{z}'}_{=1}\, \mathrm{d}\boldsymbol{\theta}' \\
&\qquad = p(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{s})\, p(\boldsymbol{\theta} \mid \boldsymbol{s}) \\
&\qquad = p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s}).
\end{aligned}$$

The same holds for a transition operator that samples $p(\boldsymbol{\theta} \mid \boldsymbol{z}, \boldsymbol{s})$, or even a single element of these sets, $p(\theta_j \mid \boldsymbol{\theta}_{\neg j}, \boldsymbol{z}, \boldsymbol{s})$. Here, θ_j is one parameter, like the transition matrix in an HMM, and $\boldsymbol{\theta}_{\neg j}$ is the set of all other parameters except for the j-th.

Many compositional models are designed such that these conditional distributions are easy to sample from. For example, if the model is defined with conjugate prior distributions, as described above, the conditional distributions have closed forms that can often be sampled exactly. Moreover, some model motifs enable more efficient types of Gibbs updates outlined below:

- Block Gibbs Sampling: In some cases, entire subsets or "blocks" of random variables can be updated by a single transition operator. Consider the conditional distribution over a

single latent state in an HMM, z_t, given all other variables,

$$p(z_t \mid \boldsymbol{s}, \boldsymbol{z}_{\neg t}, \boldsymbol{\theta}) \propto p(z_t \mid z_{t-1}, \boldsymbol{\theta})\, p(z_{t+1} \mid z_t, \boldsymbol{\theta})\, p(\boldsymbol{s}_t \mid z_t, \boldsymbol{\theta}).$$

A naïve Gibbs sampling algorithm would enumerate the K possible values of z_t, compute their posterior probability, and sample accordingly. However, this would be horribly inefficient when the states are highly correlated. Given z_{t-1} and z_{t+1}, the state z_t may essentially be deterministic. Thus, even if there is genuine uncertainty over the state sequence as a whole, this simple transition operator may get stuck in a single state sequence assignment. This would be an example of "poor mixing." To be precise, let $\widetilde{p}_\ell$ be the distribution over states of the chain after running ℓ iterations (starting from a given initial state). A Markov chain is said to mix poorly if it takes exponentially many steps before the total variation distance between the ℓ-step distribution, $\widetilde{p}_\ell$, and the true posterior distribution, p, shrinks to less than ϵ. One way to improve mixing, at least empirically, is to perform joint updates of many variables simultaneously, leveraging model-specific structure.

Consider a Gibbs step that simultaneously updates the entire state sequence of an HMM. The conditional distribution of $\boldsymbol{z}$ is proportional to the joint distribution,

$$p(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{\theta}) \propto p(z_1 \mid \boldsymbol{\theta})\, p(\boldsymbol{s}_1 \mid z_1, \boldsymbol{\theta}) \prod_{t=2}^{T} p(z_t \mid z_{t-1}, \boldsymbol{\theta})\, p(\boldsymbol{s}_t \mid z_t, \boldsymbol{\theta}).$$

While there are K^T possible assignments of $\boldsymbol{z}$, since the conditional distribution is chain structured (each state depends only on the previous state and the current spike counts), we can actually sample this distribution using dynamic programming without enumerating all possible assignments (e.g. Bishop, 2006).

- Block Parallel Gibbs Sampling: A special case of block Gibbs sampling occurs when an entire block of variables is conditionally independent given the rest. For example, consider the conditional distribution of $\boldsymbol{z}$ in a mixture model,

$$p(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{s}) \propto \prod_{t=1}^{T} p(z_t \mid \boldsymbol{\theta})\, p(\boldsymbol{s}_t \mid z_t, \boldsymbol{\theta}).$$

Since the conditional distribution factors into a product, the individual latent variables are conditionally independent of one another. That is, the update to z_t does not depend on

the updated value of $z_{t'}$. This allows us to sample new latent states in parallel using as many processors or threads as we have at our disposal.

- Collapsed Gibbs Sampling: Another special case of block Gibbs sampling occurs when the conditional distribution can be factored using the product rule. For example, consider a model with two highly correlated latent variables, z_1 and z_2. Naïvely alternating between sampling $p(z_1 \mid z_2, \boldsymbol{\theta}, \boldsymbol{s})$ and $p(z_2 \mid z_1, \boldsymbol{\theta}, \boldsymbol{s})$ will lead to poor mixing, so we would like to update them jointly. Suppose, however, that it is challenging to directly sample the full conditional distribution $p(z_1, z_2 \mid \boldsymbol{\theta}, \boldsymbol{s})$. By the sum and product rules of probability,

$$\begin{aligned} p(z_1, z_2 \mid \boldsymbol{\theta}, \boldsymbol{s}) &= p(z_2 \mid z_1, \boldsymbol{\theta}, \boldsymbol{s})\, p(z_1 \mid \boldsymbol{\theta}, \boldsymbol{s}) \\ &= p(z_2 \mid z_1, \boldsymbol{\theta}, \boldsymbol{s}) \int p(z_1, z_2 \mid \boldsymbol{\theta}, \boldsymbol{s})\, \mathrm{d}z_2. \end{aligned}$$

  If it is possible "collapse" the second variable and obtain a tractable closed form solution for $p(z_1 \mid \boldsymbol{\theta}, \boldsymbol{s})$, then we can sample the pair of variables jointly in a two step procedure. First, sample z_1 from its marginal conditional distribution, $p(z_1 \mid \boldsymbol{\theta}, \boldsymbol{s})$, and then sample $p(z_2 \mid z_1, \boldsymbol{\theta}, \boldsymbol{s})$. We use this technique in the spike-and-slab models of Chapter 5.

- Augmented Gibbs Sampling: Just as it is possible to collapse some variables during block updates, in other cases it is possible to introduce *auxiliary* variables that make the model conditionally conjugate and thus easier to work with. For example, in some cases $p(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{s})$ is challenging to sample from, but by introducing an auxiliary variable, $\boldsymbol{\omega}$, it becomes easier to sample from the conditional distributions of the full model, $p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\omega}, \boldsymbol{\theta})$. It is as if we "un-collapse" $\boldsymbol{\omega}$ and then perform augmented Gibbs sampling in two steps,

$$\begin{aligned} \boldsymbol{z}' &\sim p(\boldsymbol{z} \mid \boldsymbol{\omega}, \boldsymbol{\theta}, \boldsymbol{s}), \\ \boldsymbol{\omega}' &\sim p(\boldsymbol{\omega} \mid \boldsymbol{z}', \boldsymbol{\theta}, \boldsymbol{s}). \end{aligned}$$

  As long as the original joint distribution, $p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\theta})$, is equal to the marginal distribution, $\int p(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\omega}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\omega}$, the samples $\{\boldsymbol{z}^{(\ell)}, \boldsymbol{\theta}^{(\ell)}\}$ will be distributed according to $p(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{s})$. Thus, simply discarding the samples of $\boldsymbol{\omega}$ leaves a set of samples drawn from the desired posterior. This technique of *data augmentation* is a powerful tool that we use throughout this thesis.

2.3.2 Mean Field Variational Inference

Variational inference methods (Jordan et al., 1999; Wainwright and Jordan, 2008) take a fundamentally different approach to approximating the posterior distribution. Rather than collecting a set of samples, variational methods attempt to find the distribution within a tractable family of distributions that most closely matches the true posterior. Thus, inference becomes an optimization problem.

Let's assume the variational posterior is parameterized by $\boldsymbol{\vartheta}$, and call the variational distribution, $q(\boldsymbol{z}, \boldsymbol{\theta}; \boldsymbol{\vartheta})$. [‡] To find the optimal $q(\cdot)$, we optimize a functional, $\mathcal{L}[q]$, called the *variational lower bound*, which provides a lower bound on the log marginal likelihood, $\log p(\boldsymbol{s})$. Specifically, we can write the log marginal likelihood as an expectation with respect to q,

$$\begin{aligned}
\log p(\boldsymbol{s}) &= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log \frac{p(\boldsymbol{s},\boldsymbol{z},\boldsymbol{\theta})}{p(\boldsymbol{z},\boldsymbol{\theta}\mid\boldsymbol{s})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log \frac{p(\boldsymbol{s},\boldsymbol{z},\boldsymbol{\theta})}{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\right] + \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log \frac{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}{p(\boldsymbol{z},\boldsymbol{\theta}\mid\boldsymbol{s})}\right] \\
&= \mathcal{L}[q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})] + \mathrm{KL}(q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta}) \,||\, p(\boldsymbol{z},\boldsymbol{\theta}\mid\boldsymbol{s}) \\
&\geq \mathcal{L}[q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})].
\end{aligned}$$

where $\mathrm{KL}(q \,||\, p)$ is the KL-divergence between distributions q and p. The last line follows from the fact that the KL-divergence is nonnegative and equal to zero if and only if the distributions are identical. Thus, optimizing this functional is equivalent to minimizing the KL-divergence between the variational distribution and the true posterior.

Our goal is to maximize the variational lower bound over a parameterized family of tractable distributions, $\mathcal{Q}$. In *mean field variational inference*, we take $\mathcal{Q}$ to be the family of fully factorized distributions,

$$\mathcal{Q} = \left\{ q : q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta}) \propto \prod_{t=1}^{T} q_t(z_t;\vartheta_t) \prod_{j=1}^{J} q_j(\theta_j;\vartheta_j) \right\}.$$

We will set these parameters, $\boldsymbol{\vartheta}$, in order to maximize the variational lower bound over the set of distributions in $\mathcal{Q}$.

[‡] We use a semicolon to indicate that $q(\boldsymbol{z},\boldsymbol{\theta})$ is parameterized by $\boldsymbol{\vartheta}$. Since $\boldsymbol{\vartheta}$ is not a random variable, q is not strictly a conditional distribution and the vertical bar notation is inappropriate.

In general, this objective function is not concave, so we should not expect to find a global optimum. However, we can still use local optimization and multiple random restarts with the hope of finding a global optimum. For mean field variational inference, a simple approach is to perform coordinate ascent on the parameters of one variational factor at a time, holding the rest fixed. Given the equivalence between maximizing the variational lower bound and minimizing the KL-divergence, we can derive the general form of a mean field update. Consider updating the variational factor for θ_j. We have,

$$\begin{aligned}
\mathrm{KL}(q \,||\, p) &= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log q(\boldsymbol{z},\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log p(\boldsymbol{z},\boldsymbol{\theta} \mid \boldsymbol{s})\right] \\
&\simeq \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})}\left[\log q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})\right] - \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta}_)}\left[\log p(\boldsymbol{s},\boldsymbol{z},\boldsymbol{\theta})\right] \\
&\simeq \mathbb{E}_{q_j(\theta_j;\vartheta_j)}\left[\log q_j(\theta_j;\vartheta_j)\right] - \mathbb{E}_{q(\theta_j;\vartheta_j)}\left[\mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta}_{\neg j};\boldsymbol{\vartheta}_{\neg j})}\left[\log p(\boldsymbol{s},\boldsymbol{z},\boldsymbol{\theta})\right]\right] \\
&\simeq \mathrm{KL}\big(q_j(\theta_j;\vartheta_j) \,||\, \widetilde{p}_j(\theta_j)\big),
\end{aligned} \tag{2.8}$$

where $\simeq$ denotes equality up to an additive term that is constant with respect to θ_j, and

$$\widetilde{p}_j(\theta_j) \propto \exp\left\{\mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\theta}_{\neg j};\boldsymbol{\vartheta}_{\neg j})}\left[\log p(\boldsymbol{s},\boldsymbol{z},\boldsymbol{\theta})\right]\right\}. \tag{2.9}$$

We are able to separate the expectations in the third line of (2.8) due to the factorized form of $q(\boldsymbol{z},\boldsymbol{\theta};\boldsymbol{\vartheta})$. Since KL-divergence is minimized when the two distributions are equal, the optimal $q_j(\theta_j;\vartheta_j)$, given the variational factors for the remaining variables, is equal to $\widetilde{p}_j(\theta_j)$. As in the Gibbs sampling algorithms developed before, the expectation in (2.9) is often greatly simplified by the factorization of the joint distribution in our probabilistic model. Moreover, when the model is constructed out of conjugate distributions, these mean field updates can be computed in closed form.

Structured Mean Field Just as block Gibbs sampling enables more efficient updates for sets of correlated random variables, structured mean field algorithms allow groups of random variables to share a variational factor. For example, in an HMM, we can group the latent states $\boldsymbol{z}$ together in a shared factor, $q(\boldsymbol{z})$, that does not necessarily factor into a product over time bins. If the optimal shared factor given by (2.9) has a tractable form, we can perform coordinate ascent on the variational lower bound by updating the parameters of the shared factor, rather than sequentially updating individual factors for each time bin. As a result, our coordinate ascent algorithm will converge much more rapidly. The only other requirement is that it must be possible to compute the expectations

with respect to the shared factor. In the case of HMMs, the same type of dynamic programming algorithm that enables efficient block sampling also enables efficient calculation of expectations.

2.3.3 Model Comparison

Now that we have developed the tools to formulate models and perform Bayesian inference, we need a way to compare and criticize our models. The easiest way, and the primary way used throughout this thesis, is to compare the models based on how well they predict held-out data. Suppose that at the beginning of the experiment, we reserve a set of spike counts, $\boldsymbol{s}_{\text{test}}$, to be used for model comparison. Once we have used Bayesian inference to compute a posterior distribution over the model's parameters and latent variables, we can then compute the predictive likelihood,

$$p(\boldsymbol{s}_{\text{test}} \mid \boldsymbol{s}_{\text{train}}) = \int p(\boldsymbol{s}_{\text{test}} \mid \boldsymbol{z}_{\text{test}}, \boldsymbol{\theta})\, p(\boldsymbol{z}_{\text{test}} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \boldsymbol{s}_{\text{train}})\, \mathrm{d}\boldsymbol{z}_{\text{test}}\, \mathrm{d}\boldsymbol{\theta}. \tag{2.10}$$

Notice that this is an expectation with respect to the *posterior* distribution of $\boldsymbol{\theta}$ given the training data, and a *marginal* distribution in that it involves an integration over the latent variables associated with the test data. This integral is typically intractable, but we can construct a Monte Carlo estimate given samples from the approximate posterior,

$$p(\boldsymbol{s}_{\text{test}} \mid \boldsymbol{s}_{\text{train}}) \approx \frac{1}{L} \sum_{\ell=1}^{L} p(\boldsymbol{s}_{\text{test}} \mid \boldsymbol{z}_{\text{test}}^{(\ell)}, \boldsymbol{\theta}^{(\ell)})$$

where

$$\begin{aligned} \boldsymbol{\theta}^{(\ell)} &\sim p(\boldsymbol{\theta} \mid \boldsymbol{s}_{\text{train}}), \\ \boldsymbol{z}_{\text{test}}^{(\ell)} &\sim p(\boldsymbol{z}_{\text{test}} \mid \boldsymbol{\theta}^{(\ell)}). \end{aligned}$$

When Bayesian inference is performed with MCMC, the samples $\{\boldsymbol{\theta}^{(\ell)}\}$ are simply the states visited by the Markov chain. When variational methods are used, we assume they are drawn from the variational posterior, $q(\boldsymbol{\theta})$.

This is by no means the only method of comparing models. In "fully Bayesian" analyses, it is common to compare models on the basis of their *marginal likelihood*, $p(\boldsymbol{s})$ (Kass and Raftery, 1995). Recall that this is the quantity that variational methods attempt to lower bound. Unfortunately, we cannot evaluate the tightness of variational lower bounds because they depend on the KL-divergence, which is intractable.

Instead, we may resort to other methods of approximating the marginal likelihood. Notice that,

$$p(\boldsymbol{s}) = \int p(\boldsymbol{s} \mid \boldsymbol{z}, \boldsymbol{\theta})\, p(\boldsymbol{z} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}\, \mathrm{d}\boldsymbol{\theta} \tag{2.11}$$

is equal to the predictive likelihood in the absence of training data. Unfortunately, training data plays the crucial role of winnowing the posterior distribution over parameters. Without this constraint, simple Monte Carlo estimates like those used to approximate the predictive likelihood will suffer from extremely high variance. Instead, more sophisticated methods, like annealed importance sampling (Neal, 2001) are typically employed.

Finally, another means of evaluating and criticizing models is via *posterior predictive checks* (PPCs) (Box, 1980; Gelman et al., 2013; Blei, 2014). Though we do not make use of them in this thesis, we note that they provide a slightly different view on model performance. Rather than assessing how well the model predicts held-out data, they assess how well statistics of data simulated from the posterior distribution match statistics calculated from samples of the real data. Rather than evaluating how well one model performs relative to another, PPCs assess how well the model explains relevant aspects of the data.

2.4 Conclusion

With this background, we have the basic tools necessary to formulate models, perform Bayesian inference, and evaluate model performance. However, as we incorporate more structure into our model and scale up to larger datasets, inference quickly becomes computationally intractable. This thesis is about extending the frontier of models and motifs at our disposal by leveraging model structure to develop efficient inference algorithms. One of the major techniques we use is the introduction of auxiliary variables that render the model conjugate and enable block parallel Gibbs samplers or structured mean field algorithms. Essentially, these methods provide nice "axes" for inference. While this increases the dimensionality of the posterior, it is sometimes easier to make two simple updates rather than one hard update. These insights enable us to push the frontier of modeling and inference for complex discrete datasets like neural spike trains, and extend the set of motifs in our modeling toolkit.

3

Hawkes Processes with Latent Network Structure

Networks are fundamental models for complex systems, enabling us to reason about systems by studying the relationships between their parts. As we discussed in Chapter 1, networks are employed in a myriad of ways throughout neuroscience. Whether they represent synapses in the human connectome, population-level interactions in brain circuits, pairwise correlations in fMRI recordings, or coupling in theoretical models, networks serve as an abstraction for the messy details of systems. A network consists of a set of nodes, which may represent neurons, populations, voxels, etc., depending on the application. Connecting these nodes is a set of edges, which represent interactions between pairs of nodes, like the effect of one neuron's spikes on the subsequent activity of its downstream neighbors. By reducing a system to a network of nodes and edges, we create a simplified object for analysis.

A great deal can be learned by considering simple network properties like the average number of connections per node, or higher order statistics like "betweenness" and the number of connected triangles (Bullmore and Sporns, 2009). Some network analyses involve probabilistic modeling. For example, we may look for clusters or features of nodes that have similar patterns of connectivity. Other analyses are more supervised in nature; in a "link prediction" task we seek to predict whether or not a pair of nodes is connected, given partial observations of the network (Liben-Nowell and Kleinberg, 2007). Traditionally, network analysis has focused on *explicit network* problems in which the network itself is considered to be the observed data. That is, the nodes are and edges are con-

sidered known. A rich literature has arisen in recent years for applying statistical machine learning models to this type of problem, e.g., Liben-Nowell and Kleinberg (2007); Hoff (2008); Goldenberg et al. (2010).

In practice, however, we are often confronted with *implicit networks* that cannot be observed directly, but about which we wish to perform analysis. In an implicit network, the nodes or edges of the network may not be directly observed, but the graph structure may be inferred from noisy emissions. These noisy observations are assumed to have been generated according to an underlying model that respects the latent network structure.

For example, in connectomics problems the network must be inferred from noisy electron microscopy images; in spike train modeling the network must be inferred from weak anatomical evidence and noisy measurements of population activity; and in fMRI experiments, the network is derived from noisy blood-oxygen-level dependent (BOLD) responses. In all cases, if we knew the underlying network (i.e. if we knew how the nodes were connected), we could assign a likelihood to the observed electron microscopy images or to the measured activity. By combining this likelihood with a prior distribution that reflects our intuitions about the network structure, we construct a probabilistic model to tackle these implicit network problems.

In this chapter, we consider the case where our observations come in the form of neural spike trains, and our intuition is that a spike on one neuron will influence the activity of downstream neurons. We formalize this with a probabilistic model based on mutually interacting point processes. Specifically, we combine the Hawkes process (Hawkes, 1971) with hierarchical prior distributions on the network. This combination allows us to reason about properties of neurons that govern network structure, which in turn governs the dynamics of activity via a Hawkes process.

Figure 3.1 illustrates the components of the probabilistic model. At the highest level, we have a network of neurons. The pattern of connectivity in the network may be governed by latent variables like cell types and locations. In this case there are four types of cells (different colors in Fig. 3.1a), and each cell has a location in the two-dimensional plane. Nearby cells of the same type are more likely to connect. The network governs the dynamics of the firing rate (Fig. 3.1b), which in turn gives rise to the observed spikes (Fig. 3.1c). In this case, spikes on one neuron induce impulse responses that feed back into the firing rate of downstream neurons. These firing rate–spike train dynamics are modeled with Hawkes processes.

The rest of the chapter is organized as follows. In Section 3.1 we introduce a compositional probabilistic model for networks, and in Section 3.2 we introduce Hawkes processes. Section 3.3 stitches these two components together into a joint model for implicit networks with spike train observa-

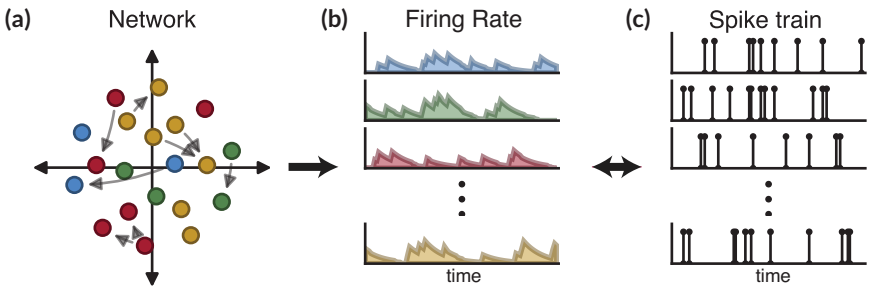

Figure 3.1: Components of the generative model. **(a)** Each neuron is endowed with latent variables, like locations in space and discrete types (illustrated with different colors). These variables determine the probability of connections and the strength of those connections. In this example, nearby neurons of the same type are most likely to connect. **(b)** The network parameterizes an autoregressive model with a time-varying firing rate, which specifies the instantaneous probability of an action potential. **(c)** Spikes are randomly generated according to the firing rate. Each spike induces an impulse response on the firing rate of downstream neurons.

tions. Before diving into inference applications, we first consider the theoretical consequences of a particular network model on the stability of the system, and provide some intuition on how the network properties affect the asymptotic behavior of the system. Then, in Section 3.4, we derive a Gibbs sampling algorithm with an elegant auxiliary variable formulation that allows efficient parallelism. Since this is the first technical chapter, we go through these derivations in substantial detail. In later chapters we will move more quickly. Finally, the remaining sections consider applications, first to synthetic data, and then to biological recordings. While the primary emphasis is on modeling neural data, we also explore some applications in areas of finance and criminology.

3.1 Probabilistic Network Models

Networks of N nodes can be represented by $N \times N$ matrices. Unweighted networks correspond to binary adjacency matrices $\boldsymbol{A}$ where $a_{m,n} = a_{m \to n} = 1$ indicates a directed edge from node m to node n. We use the arrow notation ($\to$) to remind the reader of the directionality of the connection. When these edges have scalar weights associated with them, we can encode the weights in a second matrix, $\boldsymbol{W} \in \mathbb{R}^{N \times N}$. The complete network is then defined by the elementwise product, $\boldsymbol{A} \odot \boldsymbol{W}$. The binary adjacency matrix captures the sparsity pattern, and the real-valued weight matrix captures the strength of the connections. From a modeling perspective, separating these two matrices allows us to separate our prior intuitions about sparsity and strength. This is known as a spike-and-slab model (Mitchell and Beauchamp, 1988).

Hierarchical models can be constructed by incorporating latent variables into the prior distribu-

Name	$\boldsymbol{\vartheta}$	$\mathrm{dom}(\boldsymbol{z}_n)$	$\rho_{n\to n'}$
Empty Model	—	—	0
Dense Model	—	—	1
Bernoulli Model	ρ	—	ρ
Stochastic Block Model	$\{\{\rho_{k\to k'}\}\}$	$\{1,\ldots,K\}$	$\rho_{z_n\to z_{n'}}$
Latent Distance Model	γ_0	$\mathbb{R}^K$	$\sigma(-\|\|\boldsymbol{z}_n-\boldsymbol{z}_{n'}\|\|_2^2+\gamma_0)$

Table 3.1: Binary adjacency matrix models.

tions over $\boldsymbol{A}$ and $\boldsymbol{W}$. Unsurprisingly, the same types of motifs that recur throughout probabilistic modeling — discrete latent types and continuous latent features— also form the building blocks of standard network models. We briefly outline a few simple models that are used in this and following chapters.

Table 3.1 summarizes a few models for binary adjacency matrices. In all cases, the distribution over $\boldsymbol{A}$ factorizes into a product over edges,

$$\begin{aligned} p(\boldsymbol{A}\mid \boldsymbol{z},\boldsymbol{\vartheta}) &= \prod_{n=1}^{N}\prod_{n'=1}^{N} p(a_{n\to n'}\mid \boldsymbol{z}_n,\boldsymbol{z}_{n'},\boldsymbol{\vartheta}) \\ &= \prod_{n=1}^{N}\prod_{n'=1}^{N} \mathrm{Bern}(a_{n\to n'}\mid \rho_{n\to n'}). \end{aligned}$$

The difference is in how the local latent variables, $\boldsymbol{z}_{n'}$ and $\boldsymbol{z}_n$, and the global network parameters, $\boldsymbol{\vartheta}$, combine to determine the probability, $\rho_{n\to n'}$. We describe these models below:

- *Empty Model:* The empty model is essentially a null model. According to this model, there are no connections between neurons. Nevertheless, it is useful to list it here because the empty model provides a baseline for more sophisticated models, capturing the null hypothesis that neurons are independent.

- *Dense Model:* At the other extreme, the dense model corresponds to the hypothesis that all pairs of neurons are connected. In the models of neural activity that follow, the dense model will reduce to the standard models in use today, which do not incorporate structured prior distributions over the network.

- *Bernoulli Model:* The Bernoulli model is a spike-and-slab model in which each connection

Name	$\boldsymbol{\vartheta}$	$\mathrm{dom}(\boldsymbol{z})$	$\mu_{n\to n'}$
Independent Model	μ	—	μ
Stochastic Block Model	$\{\{\mu_{k\to k'}\}\}$	$\{1,\ldots,K\}$	$\mu_{z_n\to z_{n'}}$
Latent Distance Model	μ_0	$\mathbb{R}^K$	$-\|\|\boldsymbol{z}_n-\boldsymbol{z}_{n'}\|\|_2^2+\mu_0$

Table 3.2: General weight models.

is an independent and identically distributed Bernoulli random variable. This is also known as an Erdős-Rényi model.

- *Stochastic Block Model (SBM):* In the stochastic block model (SBM) (Nowicki and Snijders, 2001), each neuron has an associated class, z_n. The probability of connection depends on the class of the two neurons. This is the network equivalent of a mixture model. In a Bayesian framework, we assume the class assignments are drawn from a discrete prior, $z_n \sim \mathrm{Discrete}(\boldsymbol{\pi})$, and the class weights are given a conjugate, symmetric Dirichlet prior, $\boldsymbol{\pi} \sim \mathrm{Dir}(\alpha \mathbf{1}_K)$. The connection probabilities are given a conjugate beta prior, $\beta_{k\to k'} \sim \mathrm{Beta}(\alpha, \beta)$.

- *Latent Distance Model:* The latent distance model (Hoff, 2008) encodes the belief that connection probability should decrease with distance between latent locations. The locations are given spherical Gaussian priors, $\boldsymbol{z}_n \sim \mathcal{N}(0, \tau \boldsymbol{I})$, and the scale is drawn from an inverse gamma prior, $\tau \sim \mathrm{IGa}(1,1)$. The offset is given a standard normal prior, $\gamma_0 \sim \mathcal{N}(0,1)$.

The same ideas can be applied to models for the scalar weight matrix, $\boldsymbol{W}$, but rather than modeling the connection probability, we now model the mean weight, $\mu_{n\to n'}$. The resulting distribution is of the form,

$$\begin{aligned} p(\boldsymbol{W} \mid \boldsymbol{z}, \boldsymbol{\vartheta}) &= \prod_{m=1}^{N} \prod_{n=1}^{N} p(w_{m\to n} \mid z_n, z_{n'}, \boldsymbol{\vartheta}) \\ &= \prod_{m=1}^{N} \prod_{n=1}^{N} p(w_{m\to n} \mid \mu_{m\to n}, \boldsymbol{\vartheta}). \end{aligned}$$

We do not specify the exact functional form of the distribution since this will depend on the model for neural activity. The linear models with nonnegative weights in this chapter will use a gamma prior, whereas the nonlinear autoregressive models in Chapter 5 will use a Gaussian distri-

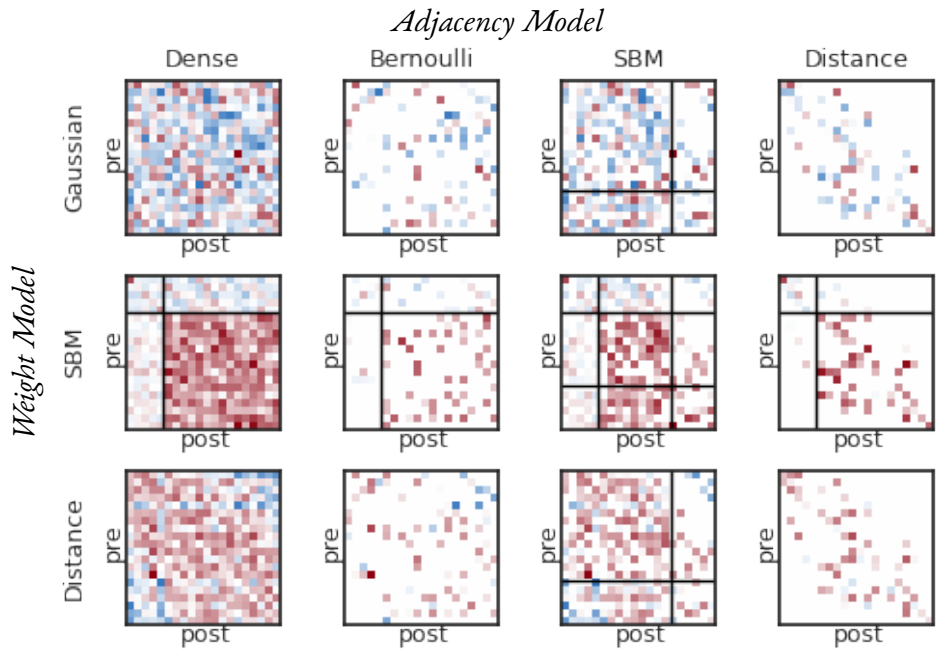

Figure 3.2: Example network models. Each row corresponds to a fixed weight matrix, $\boldsymbol{W}$, for three different weight models, and each column corresponds to a fixed adjacency matrix, $\boldsymbol{A}$, for four different adjacency models. The panels show the elementwise product of the two. Color denotes the weight (blue is negative, red is positive). In the SBM, the rows and columns are sorted by type, and in the distance model, they are sorted by location.

bution. Table 3.2 lists some examples of weight models analogous to the adjacency matrix models above. While we have only shown models for the mean weight, the same latent variables may also parameterize the variance of the weight distribution. For example in a Gaussian SBM, each directed pair of classes may have an associated variance, $\sigma^2_{k \to k'}$.

Probabilistic network models like these are unified under an elegant theoretical framework due to Aldous and Hoover (Aldous, 1981; Hoover, 1979). Conceptually, the Aldous-Hoover representation characterizes the class of *exchangeable* random graphs, that is, graph models for which the joint probability is invariant under permutations of the node labels. Just as de Finetti's theorem equates exchangeable sequences to independent draws from a random probability measure, Aldous-Hoover renders the entries of $\boldsymbol{A}$ and $\boldsymbol{W}$ conditionally independent given latent variables $\boldsymbol{z}$ and global parameters $\boldsymbol{\vartheta}$. Lloyd et al. (2012) and Orbanz and Roy (2015) review this theoretical framework and its applications in probabilistic machine learning.

Note that we have associated each neuron with a single latent variable, $\boldsymbol{z}_n$. This suggests that both the adjacency matrix and the weight matrix are governed by the same latent variable, but in

general they can have separate variables. Whether or not they are shared is a modeling decision. We will collectively refer to all latent variables of the network as $\boldsymbol{z}_n$ and whether they govern the adjacency model or the weight model will be clear from context.

Figure 3.2 shows how a variety of networks can be constructed by combining different priors on the weights (rows) with priors on the pattern of connectivity (columns). Each row corresponds to a fixed weight matrix drawn from either an independent model, a stochastic block model (SBM), or a latent distance model. In these cases, the weights are Gaussian distributed with unit variance and model-specific mean. Each column corresponds to a fixed adjacency matrix drawn from either a dense model, an independent Bernoulli model, an SBM, or a latent distance model. The matrices show the element-wise product, which encodes a weighted, directed network. Next, we introduce a model for spike trains that leverages an underlying network.

3.2 Hawkes Processes

Hawkes processes (Hawkes, 1971) are a special type of point process that allow spikes to influence the future firing rate. This is achieved via a linear superposition of Poisson processes. Before jumping into the details, a brief primer on Poisson processes is in order.

3.2.1 Poisson Processes

Point processes are fundamental statistical objects that yield random finite sets of spikes $\{s_m\}_{m=1}^M \subset \mathcal{V}$, where $\mathcal{V}$ is a compact subset of $\mathbb{R}^D$ (Daley and Vere-Jones, 2003). When modeling neural spike trains, we typically let $\mathcal{V}$ be the interval $[0, T]$. The Poisson process is the canonical example. It is governed by a nonnegative firing rate or intensity function, $\lambda(t) : \mathcal{V} \to \mathbb{R}_+$. The number of spikes in a subset $\mathcal{V}' \subset \mathcal{V}$ follows a Poisson distribution with mean $\int_{\mathcal{V}'} \lambda(t)\, \mathrm{d}t$. Moreover, the number of spikes in disjoint subsets are independent (Kingman, 1993).

We use the notation $\{s_m\}_{m=1}^M \sim \mathcal{PP}(\lambda(t))$ to indicate that a set of spikes $\{s_m\}_{m=1}^M$ is drawn from a Poisson process with rate $\lambda(t)$. There are many ways to sample a Poisson process; one way is to sample a Poisson number of spikes with mean $\int_{\mathcal{V}} \lambda(t) \mathrm{d}t$ and then sample the individual spike times, s_m, independently from the density, $p(s) = \frac{\lambda(s)}{\int_{\mathcal{V}} \lambda(t)\, \mathrm{d}t}$. Thus, after accounting for the $M!$

permutations, the likelihood of a set of spikes is given by,

$$
\begin{aligned}
p(\{s_m\}_{m=1}^M \mid \lambda(t)) &= \text{Poisson}\left(M \,\middle|\, \int_{\mathcal{V}} \lambda(t)\,\mathrm{d}t\right) \left(\prod_{m=1}^{M} \frac{\lambda(s_m)}{\int_{\mathcal{V}} \lambda(t)\,\mathrm{d}t}\right) M! \\
&= \frac{M!}{M!} \left(\int_{\mathcal{V}} \lambda(t)\,\mathrm{d}t\right)^M \exp\left\{-\int_{\mathcal{V}} \lambda(t)\,\mathrm{d}t\right\} \left(\prod_{m=1}^{M} \frac{\lambda(s_m)}{\int_{\mathcal{V}} \lambda(t)\,\mathrm{d}t}\right) \\
&= \exp\left\{-\int_{\mathcal{V}} \lambda(t)\mathrm{d}t\right\} \prod_{m=1}^{M} \lambda(s_m). \qquad (3.1)
\end{aligned}
$$

Poisson Superposition Principle We will make use of a special property of Poisson processes called the *Poisson superposition principle*, which states that if we have sets of spikes from independent Poisson processes, then the union of spikes is distributed according to a Poisson process as well (Kingman, 1993). Moreover, the rate of this process equals the sum of rates from the individual processes. Formally, suppose we are given sets of spikes drawn independently from K Poisson processes with rates $\lambda_1(t), \ldots, \lambda_K(t)$. Call the union of the spikes $\{s_m, \omega_m\}$, where $s_m \in \mathcal{V}$ is the location of the spike and $\omega_m \in \{1, \ldots, K\}$ denotes which process it came from. Let $\lambda_{\mathsf{tot}}(t) = \sum_{k=1}^K \lambda_k(t)$. The likelihood of the full set of spikes is,

$$
\begin{aligned}
p(\{s_m, \omega_m\}_{m=1}^M \mid \{\lambda_k(t)\}_{k=1}^K) &= \prod_{k=1}^{K} \mathcal{PP}(\{s_m : \omega_m = k\} \mid \lambda_k(t)) \\
&= \prod_{k=1}^{K} \left[\exp\left\{-\int_{\mathcal{V}} \lambda_k(t)\,\mathrm{d}t\right\} \prod_{m=1}^{M} \lambda_k(s_m)^{\mathbb{I}[\omega_m = k]}\right] \\
&= \exp\left\{-\int_{\mathcal{V}} \lambda_{\mathsf{tot}}(t)\,\mathrm{d}t\right\} \prod_{m=1}^{M} \prod_{k=1}^{K} \lambda_k(s_m)^{\mathbb{I}[\omega_m = k]}.
\end{aligned}
$$

The Poisson superposition principle states that the marginal distribution, summing over all pos-

sible process assignments, $\{\omega_m\}$, is a Poisson process with rate $\lambda_{\mathsf{tot}}(t)$. That is,

$$\begin{aligned} p(\{s_m\}_{m=1}^M \mid \{\lambda_k(t)\}_{k=1}^K) &= \sum_{\omega_1=1}^K \cdots \sum_{\omega_M=1}^K p(\{s_m, \omega_m\}_{m=1}^M \mid \{\lambda_k(t)\}_{k=1}^K) \\ &= \exp\left\{-\int_{\mathcal{V}} \lambda_{\mathsf{tot}}(t)\,\mathrm{d}t\right\} \prod_{m=1}^M \sum_{\omega_m=1}^K \prod_{k=1}^K \lambda_k(s_m)^{\mathbb{I}[\omega_m=k]} \\ &= \exp\left\{-\int_{\mathcal{V}} \lambda_{\mathsf{tot}}(t)\,\mathrm{d}t\right\} \prod_{m=1}^M \lambda_{\mathsf{tot}}(s_m) \\ &= \mathcal{PP}(\{s_m\} \mid \lambda_{\mathsf{tot}}(t)). \end{aligned}$$

Furthermore, the conditional distribution of ω_m is,

$$p(\omega_m = k \mid s_m, \{\lambda_k(t)\}_{k=1}^K) = \frac{\lambda_k(s_m)}{\sum_{k'=1}^K \lambda_{k'}(s_m)}.$$

In other words, given a set of spikes drawn from rate $\lambda_{\mathsf{tot}}(t)$, we can attribute each spike to one of the K additive contributions to the rate function by sampling a discrete distribution with probabilities given by the relative rate at the time of the spike. This is known as *Poisson thinning* (Kingman, 1993).

3.2.2 Including Spike History with Hawkes Processes

Though the Poisson process has many nice properties, it cannot capture interactions between spikes. For this we turn to a more general model known as the Hawkes process (Hawkes, 1971). First, consider the spike train of a single neuron, $\{s_m\}_{m=1}^M \subset [0, T]$. In a Hawkes process, the firing rate, $\lambda(t \mid \mathcal{H}_t)$, is a function of the spike history, $\mathcal{H}_t = \{s_m : s_m < t\}$. The neuron has a baseline firing rate, $\lambda^{(0)}$. On top of this baseline, each spike adds a nonnegative impulse response, $h(\Delta t)$, to the subsequent firing rate. This allows for spike-driven dynamics that are not possible in Poisson processes. Causality and locality of influence are enforced by limiting the support of $h(\Delta t)$ to $\Delta t \in [0, \Delta t_{\mathsf{max}}]$. The rate is thus given by,

$$\lambda(t \mid \mathcal{H}_t) = \lambda^{(0)} + \sum_{m=1}^M h(t - s_m).$$

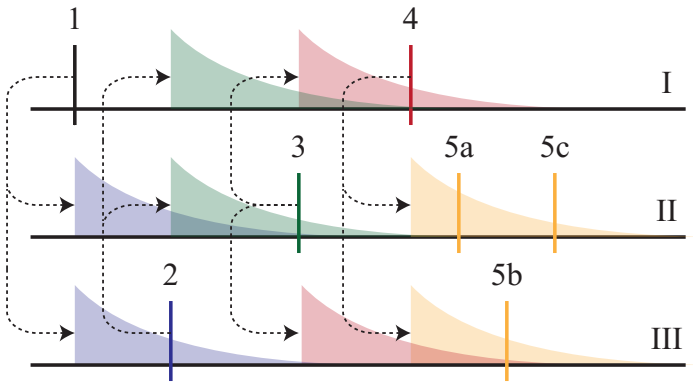

Figure 3.3: Illustration of a Hawkes process. Spikes induce impulse responses on connected processes and spawn "child" spikes. See the main text for a complete description.

When the impulse response is equal to zero, the Hawkes process reduces to a standard Poisson process with rate $\lambda^{(0)}$.

By the Poisson superposition principle, these additive components can be considered independent processes, each giving rise to their own spikes. This suggests a convenient latent variable representation in which each spike is attributed to either the background rate or the impulse response of a preceding spike. We augment our data with an auxiliary variable $\omega_m \in \{0, \ldots, m-1\}$ to indicate the *origin* of the m-th spike (0 if the spike is due to the background rate and $1 \ldots m-1$ if it was spawned by a preceding spike).

This is easily extended to a population of N neurons by considering a Hawkes process that gives rise to sets of *marked* spikes $\{s_m, c_m\}_{m=1}^M$, where $c_m \in \{1, \ldots, N\}$ specifies the neuron on which the m-th spike occurred. As in the single neuron case, the rate of the n-th neuron, $\lambda_n(t \mid \mathcal{H}_t)$, depends on the spike history, but here the spike history contains the spikes of all neurons through time t. The multi-neuronal generalization also allows for different background rates for each neuron, $\lambda_n^{(0)}$, and different impulse responses for each pair of neurons. For example, the impulse response from neuron n to neuron n', which we now call $h_{n \to n'}(\Delta t)$, may differ from that of the reverse connection. As before, we do require that the impulse responses be causal and have bounded support. Putting it all together, the rate of the n-th neuron is,

$$\lambda_n(t \mid \mathcal{H}_t) = \lambda_n^{(0)} + \sum_{m=1}^{M} h_{c_m \to n}(t - s_m). \tag{3.2}$$

After augmenting the data with auxiliary variables denoting the origin of each spike, the multi-

neuronal Hawkes process likelihood reduces to a product of Poisson process likelihoods for each background rate and each impulse response:

$$p(\{(s_m, c_m, \omega_m)\}_{m=1}^M \mid \{\lambda_n^{(0)}\}, \{\{h_{n\to n'}(\Delta t)\}\}) = \prod_{n=1}^N \mathcal{PP}\big(\{s_m : c_m = n \wedge \omega_m = 0\} \mid \lambda_n^{(0)}\big) \times \prod_{m=1}^M \prod_{n'=1}^N \mathcal{PP}\big(\{s_{m'} : c_{m'} = n' \wedge \omega_{m'} = m\} \mid h_{c_m \to n'}(t - s_m)\big).$$

Combining this with Eq. 3.1, we can write the augmented likelihood as,

$$p(\{s_m, c_m, \omega_m\}_{m=1}^M \mid \{\lambda_n^{(0)}\}, \{\{h_{n\to n'}(\Delta t)\}\}) = \prod_{n=1}^N \left[\exp\left\{-\int_0^T \lambda_n^{(0)} \mathrm{d}t\right\} \prod_{m=1}^M (\lambda_n^{(0)})^{\mathbb{I}[c_m=n]\mathbb{I}[\omega_m=0]}\right] \times \prod_{m=1}^M \prod_{n'=1}^N \left[\exp\left\{-\int_{s_m}^T h_{c_m\to n'}(t - s_m)\mathrm{d}t\right\} \prod_{m'=1}^M h_{c_m \to c_{m'}}(s_{m'} - s_m)^{\mathbb{I}[c_{m'}=n']\mathbb{I}[\omega_{m'}=m]}\right]. \quad (3.3)$$

The second line corresponds to the likelihood of the background processes; the third and fourth correspond to the likelihood of the induced processes triggered by each spike.

Figure 3.3 illustrates a causal cascades of spikes for a simple network of three processes (I-III). The first spike is caused by the background rate ($\omega_1 = 0$), and it induces impulse responses on processes II and III. Spike 2 is spawned by the impulse on the third process ($\omega_2 = 1$), and feeds back onto processes I and II. In some cases a single parent spike induces multiple children, e.g., spike 4 spawns spikes 5a-c. In this simple example, processes excite one another, but do not excite themselves.

3.3 The Network Hawkes Model

In order to combine Hawkes processes and random network models, we decompose the Hawkes impulse response $h_{n\to n'}(\Delta t)$ as follows:

$$h_{n\to n'}(\Delta t) = a_{n\to n'} \cdot w_{n\to n'} \cdot \hbar(\Delta t; \theta_{n\to n'}). \quad (3.4)$$

Here, $a_{n\to n'}$ is an entry in the binary adjacency matrix, $\boldsymbol{A} \in \{0,1\}^{N\times N}$, and $w_{n\to n'}$ is the corresponding entry in the nonnegative weight matrix, $\boldsymbol{W} \in \mathbb{R}_+^{N\times N}$. Together these specify the *sparsity structure* and *strength* of the interaction network, respectively. The nonnegative function $\hbar(\Delta t; \theta_{n\to n'})$ captures the temporal aspect of the interaction. It is parameterized by $\theta_{n\to n'}$ and satisfies two properties: a) it has bounded support for $\Delta t \in [0, \Delta t_{\mathsf{max}}]$, and b) it integrates to one. In other words, $\hbar$ is a probability density with compact support.

Decomposing the impulse response as in Equation 3.4 has many advantages. It allows us to express our separate beliefs about the sparsity structure of the interaction network and the strength of the interactions by using probabilistic network models as priors on $\boldsymbol{A}$ and $\boldsymbol{W}$. The empty graph model recovers independent background processes, and the complete graph recovers the standard Hawkes process introduced by Hawkes (1971). Making $\hbar$ a probability density endows $\boldsymbol{W}$ with units of "expected number of spikes" and allows us to compare the relative strength of interactions. The form suggests an intuitive generative model: for each impulse response draw $k \sim \text{Poisson}(w_{n\to n'})$ number of induced spikes and draw the k child spike times i.i.d. from $\hbar$. As we will see, this enables computationally tractable conjugate priors.

We can now write down the joint probability of the probabilistic model,

$$p(\{s_m, c_m, \omega_m\}, \boldsymbol{A}, \boldsymbol{W}, \{\{\theta_{n\to n'}\}\}, \{\lambda_n^{(0)}\}, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) =$$
$$p(\boldsymbol{\vartheta}) \times \overbrace{p(\{\boldsymbol{z}_n\} \mid \boldsymbol{\vartheta})}^{\text{latent variables}} \times \overbrace{p(\boldsymbol{A}, \boldsymbol{W} \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta})}^{\text{network}} \times \overbrace{p(\{\lambda_n^{(0)}\})}^{\text{background}} \times \overbrace{p(\{\theta_{n\to n'}\})}^{\text{impulses}}$$
$$\times \overbrace{p\left(\{s_m, c_m, \omega_m\} \mid \boldsymbol{A}, \boldsymbol{W}, \{\theta_{n\to n'}\}, \{\lambda_n^{(0)}\}\right)}^{\text{augmented likelihood}}. \quad (3.5)$$

Before deriving inference algorithms for this model, however, we pause to consider some of its theoretical properties.

3.3.1 Stability of Network Hawkes Processes

Due to their recurrent, mutually-excitatory nature, Hawkes processes can easily be unstable and give rise to an infinite number of spikes. A stable system must satisfy [*]

$$\lambda_{\mathsf{max}} = \max \, |\operatorname{eig}(\boldsymbol{A} \odot \boldsymbol{W})| < 1$$

[*] In this context, λ_{max} refers to an eigenvalue rather than a rate.

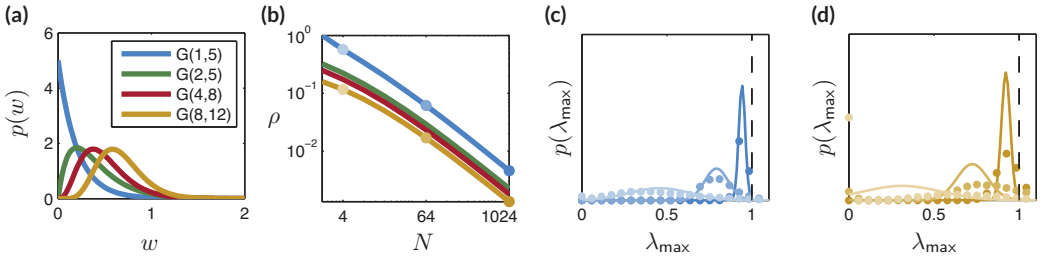

Figure 3.4: Empirical and theoretical distribution of the maximum eigenvalue for independent Bernoulli graphs with gamma weights. (a) Four gamma weight distributions. The colors correspond to the curves in the remaining panels. (b) Sparsity that theoretically yields 99% probability of stability as a function of $p(w)$ and N. (c) and (d) Theoretical (solid) and empirical (dots) distribution of the maximum eigenvalue. Color corresponds to the weight distribution in (a) and intensity indicates N and ρ shown in (b).

(c.f. Daley and Vere-Jones (2003)). For the generative model, we would like to set our hyperparameters such that the prior distribution places little mass on unstable networks. In order to do so, we use tools from random matrix theory.

The circular law describes the asymptotic eigenvalue distribution for $N \times N$ random matrices with entries that are i.i.d. with zero mean and variance σ^2. As N grows, the eigenvalues are uniformly distributed over a disk in the complex plane centered at the origin and with radius $\sigma\sqrt{N}$. In our case, however, the mean of the entries, $\mu = \mathbb{E}[a_{n\to n'} \cdot w_{n\to n'}]$, is not zero. Silverstein (1994) has analyzed such "non-central" random matrices and shown that the largest eigenvalue is asymptotically distributed as $\lambda_{\max} \sim \mathcal{N}(\mu N,\, \sigma^2)$.

In the simple case of $w_{n\to n'} \sim \mathrm{Gamma}(\kappa, \nu)$ and $a_{n\to n'} \sim \mathrm{Bern}(\rho)$, we have $\mu = \rho\kappa/\nu$ and $\sigma = \sqrt{\rho((1-\rho)\kappa^2 + \kappa)}/\nu$. We are using the rate parameterization of the gamma density,

$$\mathrm{Gamma}(w \,|\, \kappa, \nu) = \frac{\nu^\kappa}{\Gamma(\kappa)} w^{\kappa-1} e^{-\nu w}.$$

For a given N, κ and ν, we can tune the sparsity parameter ρ to achieve stability with high probability. We simply set ρ such that the minimum of $\sigma\sqrt{N}$ and, say, $\mu N + 3\sigma$, equals one. Figures 3.4a and 3.4b show a variety of weight distributions and the maximum stable ρ. Increasing the network size, the mean, or the variance will require a concomitant increase in sparsity.

This approach relies on asymptotic eigenvalue distributions, and it is unclear how quickly the spectra of random matrices will converge to this distribution. To test this, we computed the empirical eigenvalue distribution for random matrices of various size, mean, and variance. We generated 10^4 random matrices for each weight distribution in Figure 3.4a with sizes $N = 4, 64,$

and 1024, and ρ set to the theoretical maximum indicated by dots in Figure 3.4b. The theoretical and empirical distributions of the maximum eigenvalue are shown in Figures 3.4c and 3.4d. We find that for small mean and variance weights, for example $\mathrm{Gamma}(1, 5)$ in the Figure 3.4c, the empirical results closely match the theory. As the weights grow larger, as in $\mathrm{Gamma}(8, 12)$ in 3.4d, the empirical eigenvalue distributions have increased variance and lead to a greater than expected probability of unstable matrices for the range of network sizes tested here. We conclude that networks with strong weights should be counterbalanced by strong sparsity limits, or additional structure in the adjacency matrix that prohibits excitatory feedback loops.

3.4 Bayesian Inference with Gibbs Sampling

We present a Gibbs sampling procedure for inferring the model parameters, $\boldsymbol{A}$, $\boldsymbol{W}$, $\{\lambda_n^{(0)}\}$, $\{\theta_{n\to n'}\}$, and the parameters of the network, $\{\boldsymbol{z}_n\}$ and $\boldsymbol{\vartheta}$. In order to simplify our Gibbs updates, we will also sample a set of parent assignments for each spike $\{\omega_m\}$. Incorporating these parent variables enables conjugate prior distributions and a simple and efficient Gibbs sampling algorithm.

Sampling weights $\boldsymbol{W}$. To derive the updates for weights, recall from (3.4) that $w_{n\to n'}$ only appears in the impulse responses for which $c_m = n$ and $c_{m'} = n'$, so the likelihood is proportional to,

$$\begin{aligned} p(\{s_m, c_m, &\omega_m\}_{m=1}^M \mid a_{n\to n'}, w_{n\to n'}, \theta_{n\to n'}) \\ &\propto \prod_{m=1}^M \left[\exp\left\{-\int_{s_m}^T a_{n\to n'} \cdot w_{n\to n'} \cdot \hbar(t - s_m; \theta_{n\to n'})\,\mathrm{d}t\right\}\right]^{\mathbb{I}[c_m=n]} \\ &\quad \times \prod_{m=1}^M \prod_{m'=1}^M \left[w_{n\to n'}\right]^{\mathbb{I}[c_m=n]\mathbb{I}[c_{m'}=n']\mathbb{I}[\omega_{m'}=m]}. \end{aligned}$$

If $a_{n\to n'} = 0$, the impulse response is deterministically zero and, as a result, none of the spikes on neuron n' will be attributed to spikes on neuron n. Thus, the likelihood does not depend on $w_{n\to n'}$. If $a_{n\to n'} = 1$, the likelihood is more complicated. Note, however, that if $s_m < T - \Delta t_{\mathsf{max}}$,

$$\begin{aligned} -\int_{s_m}^T a_{n\to n'} \cdot w_{n\to n'} \cdot \hbar(t - s_m; \theta_{n\to n'})\,\mathrm{d}t &= -w_{n\to n'} \int_{s_m}^T \hbar(t - s_m; \theta_{n\to n'})\,\mathrm{d}t \\ &= -w_{n\to n'}, \end{aligned}$$

since $\hbar$ is a density defined on $[0, \Delta t_{\mathsf{max}}]$. In general, it is safe to ignore the impulse responses from spikes that occur in the time after $T - \Delta t_{\mathsf{max}}$ since this will be quite small compared to the total recording duration. With this approximation, the conditional distribution of $w_{n\to n'}$ reduces to,

$$p(\{s_m, c_m, \omega_m\}_{m=1}^M \mid a_{n\to n'} = 1, w_{n\to n'}) \propto e^{-M_n \cdot w_{n\to n'}} \, (w_{n\to n'})^{M_{n\to n'}}.$$

where

$$M_n = \sum_{m=1}^{M} \mathbb{I}[c_m = n],$$
$$M_{n\to n'} = \sum_{m=1}^{M} \mathbb{I}[c_m = n] \, \mathbb{I}[c_{m'} = n'] \, \mathbb{I}[\omega_{m'} = m].$$

These sufficient statistics count the number of spikes caused by an connection $n \to n'$ and the total unweighted rate induced by spikes on neuron n.

Now that we have simplified the augmented log likelihood, we see that it is conjugate with a gamma prior on the weights, $w_{n\to n'} \sim \mathrm{Gamma}(\kappa_{n\to n'}, \nu_{n\to n'})$. In Section 3.1 the weight models specified the mean, $\mu_{n\to n'}$. For a gamma distribution, $\mu_{n\to n'} = \frac{\kappa_{n\to n'}}{\nu_{n\to n'}}$. The simplest way to reconcile these is to fix the shape parameter $\kappa_{n\to n'} \equiv \kappa$, then we can compute the mean for any rate parameter, $\nu_{n\to n'}$.

Assuming κ and $\nu_{n\to n'}$ are given, the conditional distribution of the weights is,

$$p(w_{n\to n'} \mid \{s_m, c_m, \omega_m\}_{m=1}^M, a_{n\to n'} = 1, \kappa, \nu_{n\to n'}) = \mathrm{Gamma}(w_{n\to n'} \mid \widetilde{\kappa}_{n\to n'}, \widetilde{\nu}_{n\to n'}),$$

where

$$\widetilde{\kappa}_{n\to n'} = \kappa + M_{n\to n'},$$
$$\widetilde{\nu}_{n\to n'} = \nu_{n\to n'} + M_n.$$

Sampling constant background rates. Similarly, the likelihood of a constant background rate, $\lambda_n^{(0)}$, is conjugate with a gamma prior $\lambda_n^{(0)} \sim \mathrm{Gamma}(\alpha_0, \beta_0)$. The conditional distribution

is,

$$p(\lambda_n^{(0)} \mid \{s_m, c_m, \omega_m\}_{m=1}^M, \alpha_0, \beta_0) = \mathrm{Gamma}(\lambda_n^{(0)} \mid \widetilde{\alpha}_{0,n}, \widetilde{\beta}_{0,n}),$$
$$\widetilde{\alpha}_{0,n} = \alpha_0 + \sum_m \mathbb{I}[c_m = n]\, \mathbb{I}[\omega_m = 0]$$
$$\widetilde{\beta}_{0,n} = \beta_0 + T$$

Sampling impulse response parameters $\theta_{n\to n'}$. The logistic-normal density with parameters $\theta_{n\to n'} = \{\mu_{n\to n'}, \tau_{n\to n'}\}$ provides a flexible model for the impulse response:

$$\hbar(\Delta t;\, \mu_{n\to n'}, \tau_{n\to n'}) = \frac{1}{Z} \exp\left\{ \frac{-\tau_{n\to n'}}{2} \left(\sigma^{-1}\left(\frac{\Delta t}{\Delta t_{\mathsf{max}}} \right) - \mu_{n\to n'} \right)^2 \right\}$$
$$\sigma^{-1}(x) = \ln(x/(1-x))$$
$$Z = \frac{\Delta t(\Delta t_{\mathsf{max}} - \Delta t)}{\Delta t_{\mathsf{max}}} \left(\frac{\tau_{n\to n'}}{2\pi} \right)^{-\frac{1}{2}}.$$

Given the auxiliary parent variables, the likelihood is conjugate with a normal-gamma prior $\mu_{n\to n'}, \tau_{n\to n'} \sim \mathcal{NG}(\mu_\mu, \kappa_\mu, \alpha_\tau, \beta_\tau)$. The sufficient statistics are,

$$x_{m\to m'} \triangleq \ln(s_{m'} - s_m) - \ln(t_{\mathsf{max}} - (s_{m'} - s_m)),$$
$$\bar{x}_{n\to n'} = \frac{1}{M_{n\to n'}} \sum_{m=1}^M \sum_{m'=1}^M \mathbb{I}[c_m = n]\, \mathbb{I}[c_{m'} = n']\, \mathbb{I}[\omega_{m'} = m]\, x_{m\to m'},$$
$$v_{n\to n'} = \sum_{m=1}^M \sum_{m'=1}^M \mathbb{I}[c_m = n]\, \mathbb{I}[c_{m'} = n']\, \mathbb{I}[\omega_{m'} = m]\, (x_{m\to m'} - \bar{x})^2.$$

Intuitively, these correspond to the number of spikes attributed to a connection and the mean and variance of their (transformed) delays. The parameters of the normal-gamma conditional distribution are,

$$\widetilde{\mu}_{n\to n'} = \frac{\kappa_\mu \mu_\mu + M_{n\to n'} \bar{x}_{n\to n'}}{\kappa_\mu + M_{n\to n'}}, \qquad \widetilde{\kappa}_{n\to n'} = \kappa_\mu + M_{n\to n'},$$
$$\widetilde{\alpha}_{n\to n'} = \alpha_\tau + \frac{M_{n\to n'}}{2}, \qquad \widetilde{\beta}_{n\to n'} = \frac{v_{n\to n'}}{2} + \frac{M_{n\to n'}\kappa_\mu}{M_{n\to n'} + \kappa_\mu} \frac{(\bar{x}_{n\to n'} - \mu_\mu)^2}{2}.$$

Collapsed Gibbs sampling $\boldsymbol{A}$ and $\boldsymbol{\omega}$. With Aldous-Hoover graph priors, the entries in the binary adjacency matrix $\boldsymbol{A}$ are conditionally independent given the parameters of the prior. The likelihood introduces dependencies between the rows of $\boldsymbol{A}$, but each column can be sampled in parallel. This allows us to parallelize over columns and achieve an $\mathcal{O}(N)$ speedup.

Gibbs updates are complicated, however, by the strong dependencies between the graph and the parent variables. Specifically, if $\omega_{m'} = m$, then we must have $a_{c_m, c_{m'}} = 1$. To improve the performance of our sampling algorithm, first we update $\boldsymbol{A} \mid \{s_m, c_m\}, \boldsymbol{W}, \theta_{n \to n'}$ by marginalizing the parent variables. By the Poisson superposition principle, the marginal distribution is still a Poisson process:

$$\begin{aligned} p(a_{n\to n'} \mid \{s_m, c_m\}, \boldsymbol{A}_{\neg n\to n'}, \boldsymbol{W}, \boldsymbol{\theta}, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) \\ &\propto \mathcal{PP}\big(\{s_m : c_m = n'\} \,\big|\, \lambda_{n'}(t \mid \mathcal{H}_t)\big) \times p(a_{n\to n'} \mid \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta}) \\ &= \exp\left\{-\int_0^T \lambda_{n'}(t \mid \mathcal{H}_t)\,\mathrm{d}t\right\} \prod_{m=1}^{M} \left[\lambda_{n'}(s_m \mid \mathcal{H}_t)^{\mathbb{I}[c_m = n']}\right] \\ &\quad \times p(a_{n\to n'} \mid \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta}), \end{aligned}$$

where $\lambda_{n'}(t \mid \mathcal{H}_t)$ depends on $\boldsymbol{A}$, $\boldsymbol{W}$, and $\boldsymbol{\theta}$ through (3.2). Importantly, the integral of the rate function appearing in the likelihood can be computed without numerical quadrature,

$$\begin{aligned} \int_0^T \lambda_{n'}(t \mid \mathcal{H}_t)\,\mathrm{d}t &= \lambda_{n'}^{(0)} T + \sum_{m=1}^{M} a_{c_m \to n'} \cdot w_{c_m \to n'} \int_0^T \hbar(t - s_m; \theta_{c_m \to n'})\,\mathrm{d}t \\ &\approx \lambda_{n'}^{(0)} T + \sum_{m=1}^{M} a_{c_m \to n'} \cdot w_{c_m \to n'} \\ &= \lambda_{n'}^{(0)} T + \sum_{n=1}^{N} a_{n\to n'} \cdot w_{n\to n'} \cdot M_n. \end{aligned}$$

Again, the approximation stems from ignoring spikes that occur in the final interval of the recording, $(T - \Delta t_{\mathsf{max}}, T]$. For each column, we iterate over incoming edges, $a_{n\to n'}$ and sample from its collapsed distribution, holding all other parameters fixed.

Once the adjacency matrix has been updated, the parent variables are updated by Poisson thinning — that is, by sampling from their discrete conditional distribution. Again, these are all conditionally independent, so the M auxiliary variables can be sampled in parallel.

Sampling Network Variables and Parameters Given the network and the spike train, the conditional distributions for the latent variables, $\{\boldsymbol{z}_n\}$, and the parameters, $\boldsymbol{\vartheta}$ are easy by design.

- *Latent class updates*: If a stochastic block model is used for either the adjacency matrix or the weights, then it is necessary to sample the class assignments from their conditional distribution. We iterate over each neuron and update its assignment given the rest by sampling from the conditional distribution. For example, if z_n governs a stochastic block model for the adjacency matrix, the conditional distribution of the label for neuron n is given by,

$$p(z_n = k \mid \boldsymbol{z}_{\neg n}, \boldsymbol{A}, \boldsymbol{\vartheta}) \propto \pi_k \prod_{n'=1}^{N} p(a_{n' \to n} \mid \rho_{z_{n'} \to k})\, p(a_{n \to n'} \mid \rho_{k \to z_{n'}}), \tag{3.6}$$

  where $\boldsymbol{\vartheta} = \{\boldsymbol{\pi}, \{\rho_{k \to k'}\}\}$. For stochastic block models of the weight matrix, $\boldsymbol{W}$, the conditional distribution depends on $w_{n' \to n}$ and $w_{n \to n'}$ instead.

  Given the class assignments and the network, the parameters $\rho_{k \to k'}$, $\mu_{k \to k'}$, and $\boldsymbol{\pi}$ are easily updating according to their conditional distributions, since the model is conjugate.

- *Latent location updates*: We resample the locations using hybrid Monte Carlo (HMC) (Neal, 2010). Since the latent variables are continuous and unconstrained, this method is quite effective.

  In addition to the locations, the latent distance model is parameterized by a location scale, η. Given the locations and an inverse gamma prior, the inverse gamma conditional distribution can be computed in closed form.

  The remaining parameters include the log-odds, γ_0, if the distance model applies to the adjacency matrix, and the baseline mean, μ_0, if it applies to the weight matrix. These can be sampled alongside the locations with HMC.

Computational concerns. The complexity of inference is primarily driven by the number of spikes, M. We must update the auxiliary variable of each spike, and in the worst case there are m potential parents for the m-th spike. Hence, this operation can be at worst $\mathcal{O}(M^2)$ complexity. In practice, compact impulse responses limit the number of potential spike parents and significantly reduce the memory requirements and running time of our algorithm. If we could bound the maximum firing rate at λ_{max}, the complexity of resampling parent variables would

be $\sim M\, N\, \lambda_{\max}\, \Delta t_{\max}$. However, these auxiliary variables are conditionally independent so we can save a factor of M by parallelizing their updates, as we do in our parallel implementation.

The second most computationally expensive operation is updating the adjacency matrix and the weights. Note, however, that the columns of the weighted adjacency matrix are conditionally independent. Thus, we can save a factor of N by using block parallel Gibbs sampling. In order to perform these updates, we must compute sufficient statistics that involve sums over M spikes. We have implemented a multithreaded inference algorithm in Python and C that capitalizes on these opportunities for parallelism.[†]

3.5 Synthetic Results

First, we test our inference algorithm on synthetic data generated from the network Hawkes model. We perform two tests: (i) a link prediction task where the process identities are given and the goal is to simply infer whether or not an interaction exists, and (ii) a spike prediction task where we measure the probability of held-out spike sequences.

The network Hawkes model can be used for link prediction by considering the posterior probability of interactions $p(a_{n\to n'} \mid \{s_m, c_m\})$. By thresholding at varying probabilities we compute a ROC curve. A standard Hawkes process assumes a complete set of interactions ($a_{n\to n'} \equiv 1$), but we can similarly threshold its inferred weight matrix to perform link prediction.

Cross correlation provides a simple alternative measure of interaction. By binning the data and summing the cross-correlation over offsets $\Delta t \in [0, \Delta t_{\max})$, we obtain a measure of directed interaction. A probabilistic alternative is offered by the generalized linear model for point processes (GLM), a popular model for spiking dynamics in computational neuroscience (Paninski, 2004). The GLM allows for constant background rates and both excitatory and inhibitory interactions. Impulse responses are modeled with linear basis functions. Area under the impulse response provides a measure of directed excitatory interaction that we use to compute a ROC curve. In Chapter 5, we will discuss generalized linear models for spike trains in great detail.

We sampled ten network Hawkes processes of 30 nodes each with independent Bernoulli graph models, constant background rates, and the conjugate priors described above. The Hawkes processes were simulated for $T = 1000$ seconds. We used the models above to predict the presence or absence of interactions. The results of this experiment are shown in the ROC curves of Figure 3.5a. The network Hawkes model accurately identifies the sparse interactions, outperforming all other

[†] https://github.com/slinderman/pyhawkes

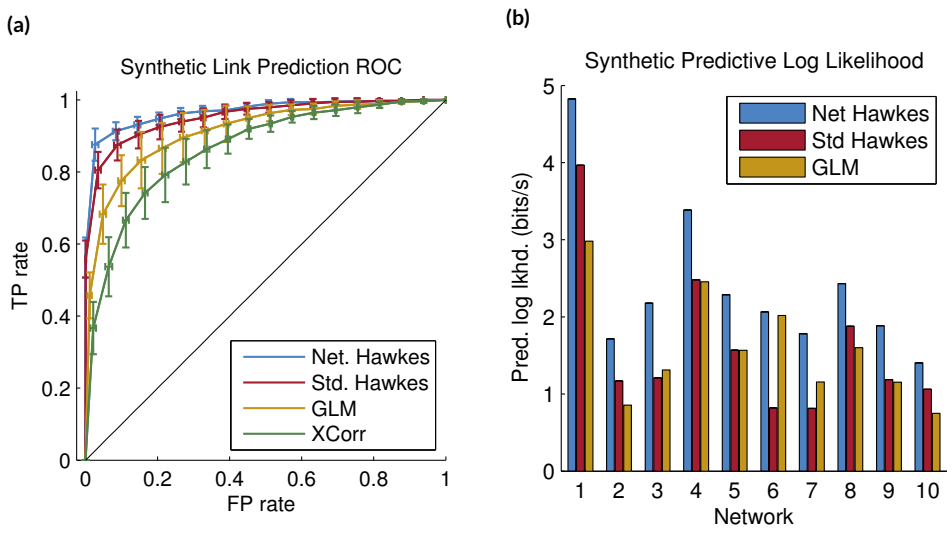

Figure 3.5: (a) Comparison of models on a link prediction test averaged across ten randomly sampled synthetic networks of 30 nodes each. The network Hawkes model with the correct independent Bernoulli graph prior outperforms a standard Hawkes model, GLM, and simple thresholding of the cross-correlation matrix. **(b)** Comparison of predictive log likelihoods, compared to a baseline of a Poisson process with constant rate. Improvement in predictive likelihood over baseline is normalized by the number of spikes in the test data to obtain units of "bits per spike." The network Hawkes model outperforms the competitors in all sample networks.

models. With the Hawkes process and the GLM we can evaluate the log likelihood of held-out test data. On this task, the network Hawkes outperforms the competitors for all networks. On average, the network Hawkes model achieves $2.2 \pm .1$ bits/spike improvement in predictive log likelihood over a homogeneous Poisson process. Figure 3.5b shows that on average the standard Hawkes and the GLM provide only 60% and 72%, respectively, of this predictive power.

3.6 Modeling Hippocampal Place Cells

Our first real dataset consists of a simultaneously recorded population of 49 hippocampal place cells from a rat freely foraging in a circular arena roughly 120cm in diameter. The data is courtesy of the lab of Prof. Matthew Wilson at MIT.[‡] The recording duration was roughly 25 minutes. The

[‡] The experiments were conducted under the supervision of the Massachusetts Institute of Technology (MIT) Committee on Animal Care and followed the NIH guidelines. The micro-drive arrays containing multiple tetrodes were implanted above the right dorsal hippocampus of male Long-Evans rats. The tetrodes were slowly lowered into the brain reaching the cell layer of CA1 two to four weeks following the date of surgery. Recorded spikes were manually clustered and sorted to obtain single units using custom software.

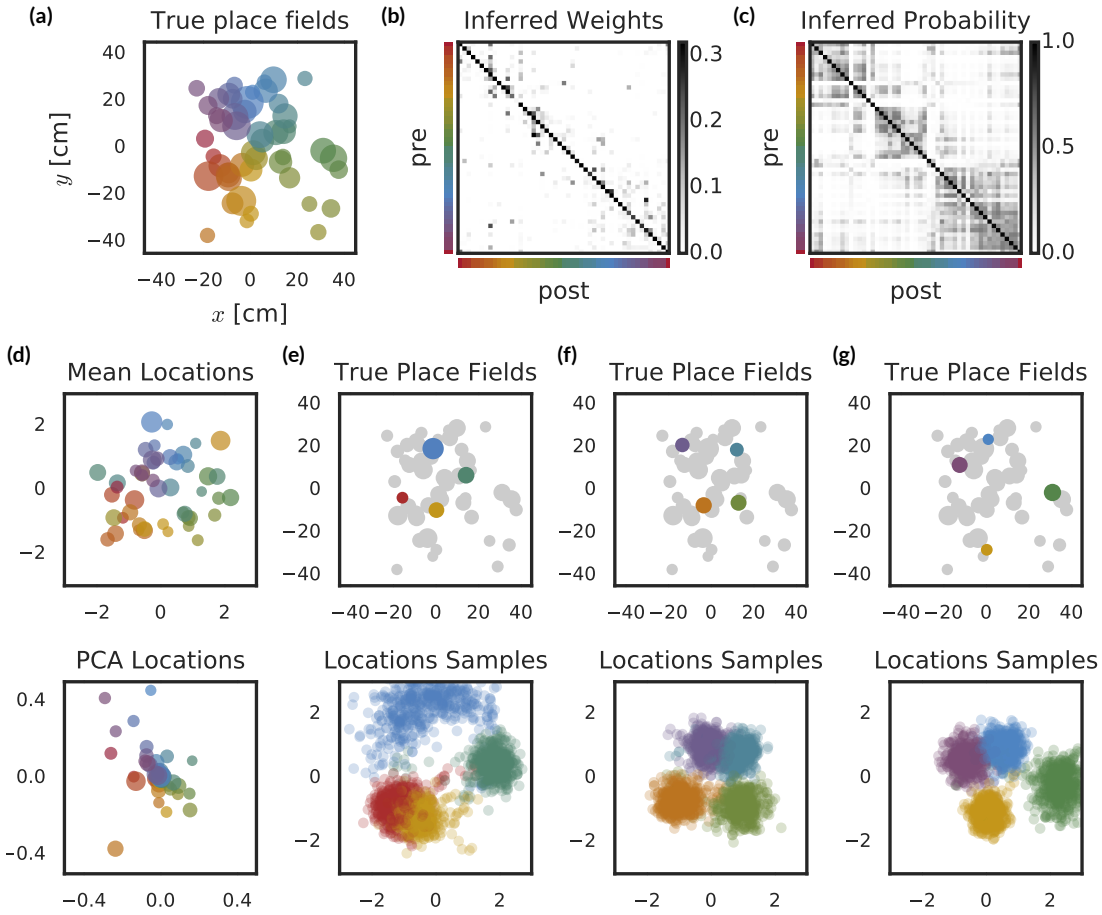

Figure 3.6: Inferred weights and locations of hippocampal place cells using a latent distance model as a prior distribution over the adjacency matrix. **(a):** True place field centers. Marker size is proportional to the size of the place field. Neurons are false colored for identification. **(b):** Expected weights under posterior. Neurons are sorted by location, and colorbars on x and y axes map to colors in **(a)**. **(c):** Expected connection probability under posterior according to latent distance model. **(d):** Mean posterior locations under the latent distance model. For comparison, the embedding found by PCA is plotted below. **(e-g):** Posterior distribution over locations shown four cells at a time. **top:** True locations of the cells. **bottom:** 250 samples from the posterior distribution over neuron locations for the four cells colored above.

first 20 minutes were used for model fitting and the last 5 were reserved for predictive tests. Over the entire 25 minute recording, each neuron fired on average 1979 ± 4117 spikes (min: 48, max: 27572) for a total of 97000 spikes. This corresponds to a firing rate of 1.35 ± 2.82Hz (min: 0.32Hz, max: 18.88Hz). The rat's location, $\boldsymbol{x}(t)$, was recorded along with the corresponding spike times.

From this, the place field of the n-th neuron is computed as,

$$\bar{\boldsymbol{x}}_n = \frac{1}{M_n} \sum_{m=1}^{M} \boldsymbol{x}(s_m) \cdot \mathbb{I}[c_m = n],$$

where, again, M_n is the number of spikes fired by neuron n. Likewise, the covariance of the place field is given by,

$$\boldsymbol{V}_n = \left[\frac{1}{M_n} \sum_{m=1}^{M} \boldsymbol{x}(s_m)\boldsymbol{x}(s_m)^\mathsf{T} \cdot \mathbb{I}[c_m = n] \right] - \bar{\boldsymbol{x}}_n \bar{\boldsymbol{x}}_n^\mathsf{T}.$$

This gives us an estimate of the size of the place field. Let $\boldsymbol{X} = \{\boldsymbol{x}_n\}$ denote the 49×2 matrix of place fields.

We compare a few different models for this data. Our baseline is a set of independent Poisson processes with constant firing rates set by maximum likelihood. Next, we consider a standard, densely connected Hawkes process. Third, we fit a network Hawkes process with an independent Bernoulli prior on the adjacency matrix and an independent gamma prior on the weight matrix. This induces sparsity in the connectivity. Finally, we fit a network Hawkes model with a latent distance prior on the adjacency matrix and an independent gamma prior on the weights. In fitting this last model, we infer a distribution over sets of locations, $\boldsymbol{Z} = \{\boldsymbol{z}_n\}$, for the population. Intuitively, we expect these locations to mirror the true place fields since nearby cells are likely to have correlated firing rates, which should be captured by excitatory impulse responses between nearby cells.

Figure 3.6 shows the posterior distribution from the network Hawkes model with a latent distance model prior. Figure 3.6a shows the true place fields of the 49 neurons. The marker size is proportional to the size of the place field, as measured by the largest eigenvalue of $\boldsymbol{V}_n$. The neurons are false colored for identification. Figure 3.6b and 3.6c show the expected weights, $\mathbb{E}[\boldsymbol{W}]$, and the matrix of expected connection probabilities, $\mathbb{E}[\rho_{n \to n'}]$, respectively. The colorbars on the axes map to colors in Figure 3.6a. Nearby neurons have higher probability of connection, as expected. This is reflected in the inferred locations.

Since the latent distance model is invariant to rotation, for each sample $\boldsymbol{Z}^{(\ell)}$, we find the orthogonal matrix, $\boldsymbol{R}^{(\ell)}$, that minimizes $||\boldsymbol{X} - \boldsymbol{R}^{(\ell)}\boldsymbol{Z}^{(\ell)}||_F$ and apply it to obtain a rotated set of locations, $\widetilde{\boldsymbol{Z}}^{(\ell)} = \boldsymbol{R}^{(\ell)}\boldsymbol{Z}^{(\ell)}$. Doing this for each sample yields a set of locations $\{\widetilde{\boldsymbol{Z}}^{(\ell)}\}_{\ell=1}^{L}$. Figure 3.6d (top) shows the mean posterior locations, $\mathbb{E}[\widetilde{\boldsymbol{Z}}]$, and we see that it is qualitatively very similar to the

Hippocampal Model	Pred. log lkhd. (bits/spike)
Standard Hawkes	$0.750 \pm 9.7 \times 10^{-5}$
Net. Hawkes ($\boldsymbol{A} \sim$ Bernoulli Model)	$0.768 \pm 9.2 \times 10^{-5}$
Net. Hawkes ($\boldsymbol{A} \sim$ Latent Distance Model)	$0.766 \pm 9.4 \times 10^{-5}$

Table 3.3: Comparison of hippocampal models on a spike prediction task relative to a homogeneous Poisson process baseline.

true locations. In contrast, the two dimensional PCA embedding is highly skewed.[§]

Finally, panels (e-g) show the samples from the posterior distribution of $\widetilde{\boldsymbol{Z}}$. Since this is difficult to visualize, we show the marginal distribution of four neurons at a time. The true location of the four place fields is identified in the upper panel, and the sampled locations are scattered in the lower panel. Importantly, the relative arrangement of locations is well preserved in the inferred locations. Moreover, cells with larger place fields tend to have larger posterior variance in their locations. This is to be expected since large place fields imply imprecise coding of space and higher correlation with other cells. This illustrates how Hawkes processes combined with latent variable models can provide interpretable portraits of complex datasets and find low-dimensional embeddings that recover intuitive structure.

Table 3.3 lists the predictive likelihoods of the various models relative to a homogeneous Poisson process baseline in units of bits per spike. We see that the sparsity of the network Hawkes model leads to improved predictive performance. In this case, however, the simple independent Bernoulli model and the latent distance model both have similar predictive likelihoods. When the network is largely determined by the training data, the prior has little effect. Thus, the two sparse priors may yield similar predictive performance, even though the latent distance model identifies meaningful latent structure. We will consider additional ways of disambiguating different network models in Chapter 5.

3.7 Trades on the S&P 100

While the focus of this thesis is on modeling *neural* spike trains, these models have broad applicability outside neuroscience as well. Here we present one example in which we study the trades on the

[§]First, we binned the spikes into 250ms bins, then we smoothed the spike counts with a Gaussian kernel of width 1s to estimate the firing rate. Finally, we applied PCA to the firing rate matrix and used the top two principal components as the embedding.

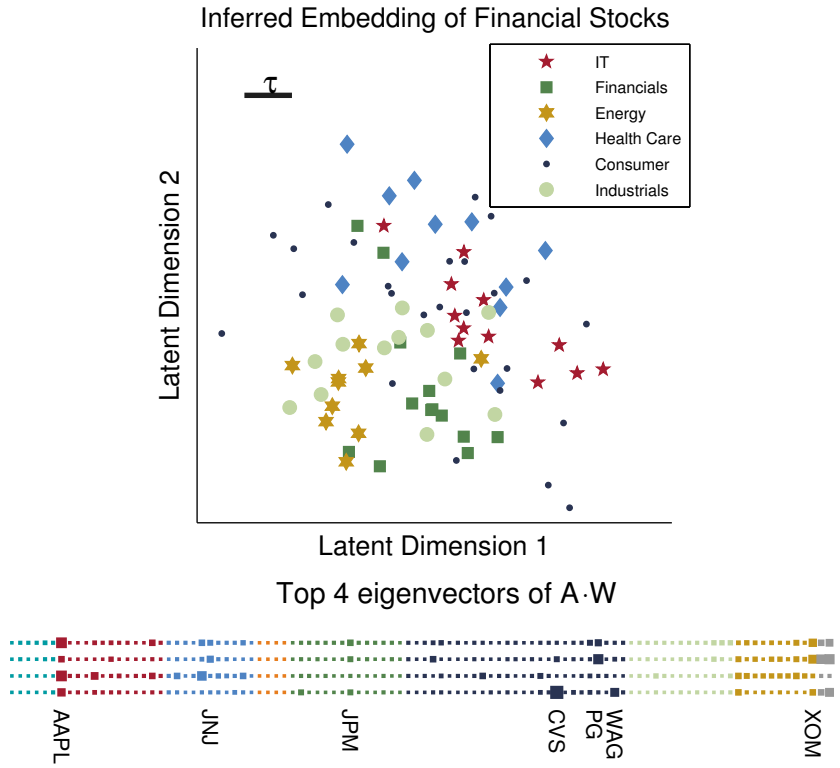

Figure 3.7: Top: A sample from the posterior distribution over embeddings of stocks from the six largest sectors of the S&P100 under a latent distance graph model with two latent dimensions. Scale bar: the characteristic length scale of the latent distance model. The latent embedding tends to embed stocks such that they are nearby to, and hence more likely to interact with, others in their sector. **Bottom**: Hinton diagram of the top 4 eigenvectors. Size indicates magnitude of each stock's component in the eigenvector and colors denote sectors as in the top panel, with the addition of Materials (aqua), Utilities (orange), and Telecomm (gray). We show the eigenvectors corresponding to the four largest eigenvalues $\lambda_{\text{max}} = 0.74$ (top row) to $\lambda_4 = 0.34$ (bottom row).

S&P 100 index collected at 1s intervals during the week of Sep. 28 through Oct. 2, 2009. Every time a stock price changes by $\pm 0.1\%$ of its current price a spike is logged on the stock's process, yielding a total of $N = 100$ processes and $M = 182,037$ spikes.

Trading volume varies substantially over the course of the day, with peaks at the opening and closing of the market. Rather than attempting to model this background fluctuation with a constant background rate, here we use a log Gaussian Cox process (LGCP) (Møller et al., 1998) with a periodic kernel instead. Complete details of inference are given in Linderman and Adams (2014). We look for short-term interactions on top of this background rate with time scales of $\Delta t_{\text{max}} = 60$s.

In Figure 3.4 we compare the predictive performance of independent LGCPs, a standard Hawkes

Financial Model	Pred. log lkhd. (bits/spike)
Independent LGCP	0.594
Standard Hawkes	0.912
Net. Hawkes ($\boldsymbol{A} \sim$ Bernoulli Model)	0.903
Net. Hawkes ($\boldsymbol{A} \sim$ Latent Distance Model)	0.888

Table 3.4: Comparison of financial models on a spike prediction task, relative to a homogeneous Poisson process baseline.

process with LGCP background rates, and the network Hawkes model with LGCP background rates under two graph priors. The models are trained on four days of data and tested on the fifth. Though the network Hawkes is slightly outperformed by the standard Hawkes, the difference is small relative to the performance improvement from considering interactions, and the inferred network parameters provide interpretable insight into the market structure.

In the latent distance model for $\boldsymbol{A}$, each stock has a latent embedding $\boldsymbol{z}_n \in \mathbb{R}^2$ such that nearby stocks are more likely to interact, as described in Section 3.1. Figure 3.7 shows a sample from the posterior distribution over embeddings in $\mathbb{R}^2$. We have plotted stocks in the six largest sectors, as listed on Bloomberg.com. Some sectors, notably energy and financials, tend to cluster together, indicating an increased probability of interaction between stocks in the same sector. Other sectors, such as consumer goods, are broadly distributed, suggesting that these stocks are less influenced by others in their sector. For the consumer industry, which is driven by slowly varying factors like inventory, this may not be surprising.

The Hinton diagram in the bottom panel of Figure 3.7 shows the top 4 eigenvectors of the interaction network. All eigenvalues are less than 1, indicating that the system is stable. The top row corresponds to first eigenvector ($\lambda_{\text{max}} = 0.74$). Apple (`AAPL`), J.P. Morgan (`JPM`), and Exxon Mobil (`XOM`) have notably large entries in the eigenvector, suggesting that their activity will spawn cascades of self-excitation.

3.8 Gangs of Chicago

As a second example of applications outside neuroscience, we study spatiotemporal patterns of gang-related homicide in Chicago. Sociologists have suggested that gang-related homicide is mediated by underlying social networks and occurs in mutually-exciting, retaliatory patterns (Papachristos, 2009). This is consistent with a spatiotemporal Hawkes process in which processes correspond

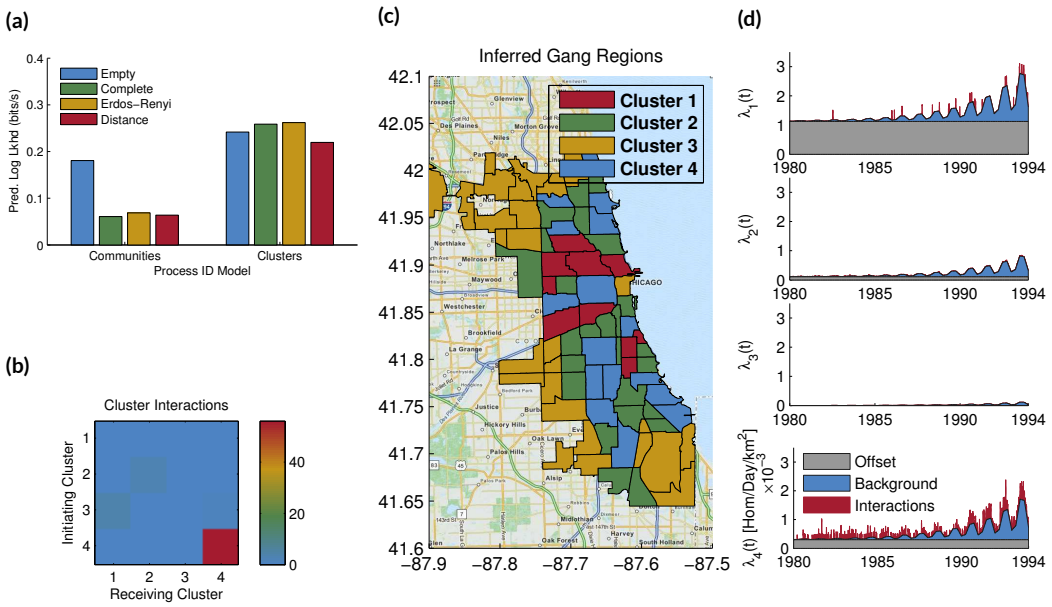

Figure 3.8: Inferred interactions among clusters of community areas in the city of Chicago. **(a)** Predictive log likelihood for "communities" and "clusters" process identity models and four graph models. Panels **(b-d)** present results for the model with the highest predictive log likelihood: an independent Bernoulli graph with $N = 4$ clusters. **(b)** The weighted interaction network in units of induced homicides over the training period (1980-1993). **(c)** Inferred clustering of the 77 community areas. **(d)** The intensity for each cluster, broken down into the offset, the shared background rate, and the interactions (units of 10^{-3} homicides per day per square kilometer).

to gang territories and homicides incite further homicides in rival territories.

We study gang-related homicides between 1980 and 1995 (Block et al., 2005). Homicides are labeled by the community in which they occurred. Over this time-frame there were $M = 1637$ gang-related homicides in the 77 communities of Chicago.

We evaluate our model with a spike-prediction task, training on 1980-1993 and testing on 1994-1995. We use a LGCP temporal background rate in all model variations. Our baseline is a single process with a uniform spatial rate for the city. Here, however, the analogous "neurons" are not so clear. We consider two models: (i) the "community" model, which considers each community a separate "neuron," or process, and (ii) the "cluster" model, which groups communities into processes. The number of clusters is chosen by cross-validation (again, see Linderman and Adams (2014)). For each process identity model, we compare four graph models: (i) independent LGCPs (*empty*), (ii) a standard Hawkes process with all possible interactions (*complete*), (iii) a network Hawkes model with a sparsity-inducing independent Bernoulli graph prior, and (iv) a network Hawkes model with a

latent distance model that prefers short-range interactions.

The community process identity model improves predictive performance by accounting for higher rates in South and West Chicago where gangs are deeply entrenched. Allowing for interactions between community areas, however, results in a decrease in predictive power due to overfitting (there is insufficient data to fit all 77^2 potential interactions). Interestingly, sparse graph priors do not help. They bias the model toward sparser but stronger interactions which are not supported by the test data. These results are shown in the "communities" group of Figure 3.8a. Clustering the communities improves predictive performance for all graph models, as seen in the "clusters" group. Moreover, the clustered models benefit from the inclusion of excitatory interactions, with the highest predictive log likelihoods coming from a four-cluster independent Bernoulli graph model with interactions shown in Figure 3.8b. Distance-dependent graph priors do not improve predictive performance on this dataset, suggesting that either interactions do not occur over short distances, or that local rivalries are not substantial enough to be discovered in our dataset. More data is necessary to conclusively say which.

Looking into the inferred clusters in Figure 3.8c and their rates in 3.8d, we can interpret the clusters as "safe suburbs" in gold, "buffer neighborhoods" in green, and "gang territories" in red and blue. Self-excitation in the blue cluster (Figure 3.8b) suggests that these regions are prone to bursts of activity, as one might expect during a turf-war. This interpretation is supported by reports of "a burst of street-gang violence in 1990 and 1991" in West Englewood ($41.77°$N, $-87.67°$W) (Block and Block, 1993).

Figure 3.8d also shows a significant increase in the homicide rate between 1989 and 1995, consistent with reports of escalating gang warfare (Block and Block, 1993). In addition to this long-term trend, homicide rates show a pronounced seasonal effect, peaking in the summer and tapering in the winter. An LGCP with a quadratic kernel point-wise added to a periodic kernel captures both effects.

3.9 Related Work

Hawkes processes and latent network discovery have been a subject of recent interest in the machine learning community. Much of this interest stems from the growth of social networking applications which produce massive amounts of spiking data. Gomez-Rodriguez et al. (2010) introduced one of the earliest algorithms for discovering latent networks from cascades of spikes in social network data. They developed a highly scalable approximate inference algorithm, but they did not explore

the potential of random network models or emphasize the point process nature of the data. Simma and Jordan (2010) studied this problem from the context of Hawkes processes and developed an expectation-maximization inference algorithm that could scale to massive datasets, like the interactions between authors on Wikipedia. We have adapted their latent variable formulation in our fully-Bayesian inference algorithm and introduced a framework for prior distributions over the latent network.

Others have considered special cases of the model we have proposed. Blundell et al. (2012) combine Hawkes processes and the Infinite Relational Model (a specific exchangeable graph model with an Aldous-Hoover representation) to cluster processes and discover interactions in email networks. Cho et al. (2013) applied Hawkes processes to gang incidents in Los Angeles. They developed a spatial Gaussian mixture model (GMM) for process identities, but did not explore structured network priors. We experimented with this process identity model but found that it suffers in predictive log likelihood tests.

Iwata et al. (2013) developed a stochastic EM algorithm for Hawkes processes, leveraging similar conjugacy properties, but without network priors. Zhou et al. (2013) have developed a promising optimization-based approach to discovering low-rank networks in Hawkes processes, similar to some of the network models we explored. Guo et al. (2014) have developed a similar model to ours. They focus on applying Hawkes processes to language modeling and incorporating features of the discrete events. DuBois et al. (2013) also explored the use of infinite relational models as a prior in conjunction with a point process observation model build on a Gibbs sampling algorithm.

Perry and Wolfe (2013) derived a partial likelihood inference algorithm for Hawkes processes with a similar emphasis on structural patterns in the network of interactions. They provide an estimator capable of discovering homophily and other network effects. Our fully-Bayesian approach generalizes this method to capitalize on recent developments in random network models (Lloyd et al., 2012).

Finally, generalized linear models (GLMs) are widely used in computational neuroscience (Paninski, 2004). GLMs allow for both excitatory and inhibitory interactions, but, as we have shown, when the data consists of purely excitatory interactions, Hawkes processes outperform GLMs in link- and spike-prediction tests. We will discuss these models in Chapter 5

3.10 Conclusion

This chapter developed a framework for discovering latent network structure from spiking data with mutually excitatory interactions. Our auxiliary variable formulation of the multivariate Hawkes process supports a broad class of prior distributions on latent network structure. This allows us to connect interpretable latent variables, like neuron types and features, to a dynamic model for spike trains. Our parallel MCMC algorithm allowed us to reason about uncertainty in the latent network in a fully-Bayesian manner. We leveraged results from random matrix theory to analyze the conditions under which random network models will be stable, and our applications uncovered interpretable latent networks in a variety of synthetic and real-world problems.

Hawkes processes are the point process analogue of linear autoregressive models. The firing rate is a sum of nonnegative impulse responses induced by preceding spikes. As we generalize these models in the following chapters, we will exploit this relationship and consider natural extensions like discrete time, nonlinear, and nonstationary versions of the model.

4

Discrete-Time Linear Autoregressive Poisson Models

This chapter builds on the network Hawkes model introduced in the Chapter 3. We introduce linear autoregressive Poisson models — the discrete time analogue of the Hawkes process — and we derive efficient Gibbs sampling and stochastic variational inference algorithms, leveraging the Poisson superposition principle as before. This chapter marks the transition from continuous time models to the discrete time models that occupy this and subsequent chapters. As we will see, these discrete time formulations are in some ways easier to work with. We can easily extend them to non-Poisson spike count models, and we can interface with a diverse array of probabilistic matrix decomposition models. However, the discrete nature of spike counts still poses some serious inferential hurdles, which this thesis aims to overcome.

In addition to bridging from continuous to discrete, this chapter also addresses issues of computational complexity. The complexity of our Hawkes process inference algorithm scaled, in the worst case, quadratically with the number of spikes, since we had to sample a "parent" for each spike. By designing block parallel Gibbs updates, we were able to obtain linear complexity in the number of spikes. However, when the firing rates are high, this is still the bottleneck of our algorithm. Here, we reduce this complexity to be independent of the number of spikes by adopting a discrete time approach. Moreover, we derive efficient *stochastic* variational inference algorithms (Hoffman et al., 2013) that work with subsets of time bins in each iteration and thereby scale to massive datasets.

4.1 Probabilistic Model

The fundamental limitation of the previously developed continuous time models is that the domain of the auxiliary variable, ω_m, grows with the number of events which occurred before time s_m. For datasets with high rates of activity, this can quickly become the limiting factor of the inference algorithm. At the same time, it is often reasonable to assume that events do not interact on time scales shorter than Δt. This motivates a discrete time formulation in which we group events into bins of width Δt and ignore potential interactions between events in the same bin. Then the rate becomes,

$$\begin{aligned} \lambda_{t,n} &= \lambda_n^{(0)} + \sum_{n'=1}^{N} \sum_{d=1}^{D} s_{t-d,n'} \cdot h_{n' \to n}[d], \\ s_{t,n} &\sim \text{Poisson}(\lambda_{t,n} \cdot \Delta t), \end{aligned} \tag{4.1}$$

where $s_{t,n}$ is the number of spikes fired by neuron n in the t-th time bin and $h_{n' \to n}[d]$ is an impulse response function describing the influence that events on neuron n' have on the rate of process n at discrete time lag d. As we will show, under this formulation the auxiliary variables only assume a fixed set of values independent of the rate.

As in the last chapter, we introduce a network model as a prior distribution over the impulse response weights. Following the approach of the previous chapter, we decompose the impulse response function into the product of a binary variable that specifies whether or not a connection exists, a scalar weight that specifies the strength of the interaction if present, and a probability mass function that specifies the time course of interaction:

$$h_{n \to n'}[d] = a_{n \to n'} \cdot w_{n \to n'} \cdot \hbar[d; \boldsymbol{\theta}_{n \to n'}]$$

for $d \in \{1, \ldots, D\}$. The function $\hbar[d] : \{1, \ldots, D\} \to [0, 1]$ is now a probability mass function,

which we model as a convex combination of normalized basis functions, $\boldsymbol{\phi}_b$,

$$\begin{aligned}\hbar[d;\, \boldsymbol{\theta}_{n\to n'}] &\triangleq \sum_{b=1}^{B} \theta^{(b)}_{n\to n'} \cdot \phi_b[d], \\ \sum_{d=1}^{D} \phi_b[d] \cdot \Delta t &= 1, \\ \sum_{b=1}^{B} \theta^{(b)}_{n\to n'} &= 1.\end{aligned}$$

We enforce the latter constraint with a Dirichlet prior $\boldsymbol{\theta}_{n\to n'} \sim \mathrm{Dir}(\boldsymbol{\gamma})$. The basis functions are typically taken to be normalized Gaussian bumps or rectified cosine functions spaced over the interval $1, \ldots, D$. For example, in the following experiments we used,

$$\begin{aligned}\widetilde{\phi}_b[d] &= \exp\left\{-\frac{1}{2\sigma^2}(d-\mu_b)^2\right\}, \\ \phi_b[d] &= \frac{\widetilde{\phi}_b[d]}{\Delta t \sum_{d'=1}^{D} \widetilde{\phi}_b[d']},\end{aligned}$$

with means, μ_b, evenly spaced on $[1, D]$, and $\sigma = \frac{D}{B-1}$.

Plugging this impulse response model into Eq. 4.1 yields,

$$\begin{aligned}\lambda_{t,n'} &= \lambda^{(0)}_{n'} + \sum_{n=1}^{N}\sum_{b=1}^{B} a_{n\to n'} \cdot w_{n\to n'} \cdot \theta^{(b)}_{n\to n'} \sum_{t'=1}^{t-1} s_{t',n} \cdot \phi_b[d] \\ &= \lambda^{(0)}_{n'} + \sum_{n'=1}^{N}\sum_{b=1}^{B} a_{n\to n'} \cdot w_{n\to n'} \cdot \theta^{(b)}_{n\to n'} \cdot \widehat{s}_{t,n,b},\end{aligned}$$

where

$$\widehat{s}_{t,n,b} \triangleq (\boldsymbol{s}_n * \boldsymbol{\phi}_b)\,[t]$$

is the discrete convolution of the n-th spike train with the b-th basis function evaluated at the t-th time bin. Since both the spike trains and the basis functions are given, these can be precomputed.

As before, we begin by introducing auxiliary parent variables for each entry $s_{t,n}$. By the superposition theorem for Poisson processes, each event can be attributed to either the background rate or one of the impulse responses.

Let $\omega_{t,n'}^{(n,b)} \in \{0, \dots, s_{t,n'}\}$ denote how many of the events that occurred in the t-th time bin on the n'-th neuron are attributed to the b-th basis function of the n-th neuron. Similarly, let $\omega_{t,n'}^{(0)}$ denote the number of events attributed to the background process. We combine these auxiliary variables into vectors, $\boldsymbol{\omega}_{t,n'} \triangleq \left[\omega_{t,n'}^{(0)}, \omega_{t,n'}^{(1,1)}, \dots, \omega_{t,n'}^{(N,B)}\right]$.

Due to the Poisson superposition principle, these parent variables are conditionally multinomial distributed. For time t and neuron n', we resample

$$\boldsymbol{\omega}_{t,n'} \sim \text{Mult}\left(s_{t,n'}, \boldsymbol{u}_{t,n'}\right) \quad u_{t,n'}^{(0)} = \frac{\lambda_{n'}^{(0)}[t]}{\lambda_{n'}[t]}, \quad u_{t,n'}^{(n,b)} = \frac{\hat{s}_{t,n,b} \cdot a_{n\to n'} \cdot w_{n\to n'} \cdot \theta_{n\to n'}^{(b)}}{\lambda_{n'}[t]}.$$

Given this attribution, the likelihood factorizes into a product of Poisson distributions,

$$p(\boldsymbol{\omega} \mid \boldsymbol{\lambda}) = \left[\prod_{t=1}^{T}\prod_{n'=1}^{N} \text{Poisson}(\omega_{t,n'}^{(0)} \mid \lambda_{n'}^{(0)}\Delta t)\right] \times \left[\prod_{t=1}^{T}\prod_{n=1}^{N}\prod_{n'=1}^{N}\prod_{b=1}^{B} \text{Poisson}(\omega_{t,n'}^{(n,b)} \mid \hat{s}_{t,n,b} \cdot a_{n\to n'} \cdot w_{n\to n'} \cdot \theta_{n\to n'}^{(b)} \cdot \Delta t)\right].$$

Gibbs sampling the background rates. We use conjugate priors for the constant background rates, weights, and impulse responses. For the constant background rates we have, $\lambda_{n'}^{(0)} \sim \text{Gamma}(\alpha_\lambda, \beta_\lambda)$, which results in the conditional distribution

$$\begin{aligned}\lambda_{n'}^{(0)} \mid \{\omega_{t,n'}^{(0)}\} &\sim \text{Gamma}(\alpha_\lambda^{(n)}, \beta_\lambda^{(n)}),\\ \alpha_\lambda^{(n)} &= \alpha_\lambda + \sum_{t=1}^{T} \omega_{t,n'}^{(0)},\\ \beta_\lambda^{(n)} &= \beta_\lambda + T\Delta t\,.\end{aligned}$$

GIBBS SAMPLING IMPULSE RESPONSES. The likelihood of the impulse responses, $\boldsymbol{\theta}_{n\to n'}$ is proportional to a Dirichlet distribution. Combined with a Dirichlet($\boldsymbol{\gamma}$) prior this yields

$$\boldsymbol{\theta}_{n\to n'} \mid \{\omega_{t,n}^{(n',b)}\}, \boldsymbol{\gamma} \sim \mathrm{Dir}\left(\boldsymbol{\gamma}_{n\to n'}\right),$$
$$\gamma_{n\to n'}^{(b)} = \gamma_b + \sum_{t=1}^{T} \omega_{t,n'}^{(n,b)}.$$

GIBBS SAMPLING THE WEIGHTED ADJACENCY MATRIX. As before, the weights are conjugate with a gamma prior, $\mathrm{Gamma}(\kappa, \nu_{n\to n'})$, where the scale is presumed to be given by the network prior. Given the adjacency matrix $\boldsymbol{A}$ and the auxiliary parent variables, the conditional distribution is,

$$w_{n\to n'} \mid a_{n\to n'} = 1 \sim \mathrm{Gamma}(\widetilde{\kappa}^{(n,n')}, \widetilde{\nu}^{(n,n')}),$$
$$\widetilde{\kappa}^{(n,n')} = \kappa + \sum_{t=1}^{T}\sum_{b=1}^{B} \omega_{t,n'}^{(n,b)},$$
$$\widetilde{\nu}^{(n,n')} = \nu_{n\to n'} + \sum_{t=1}^{T} s_{t,n}.$$

As in the previous chapter, in order to resample $\boldsymbol{A}$, we iterate over each entry and sample from the conditional distribution after integrating out the parents. We assume the parameters of the network prior can be sampled efficiently — a reasonable assumption for many exchangeable random network models.

The continuous time representation introduces a latent "parent" variable for each event in the dataset, and the parent can be any one of the events that occurred in the preceding window of influence. Call the number of potential parents M. The discrete time representation has a multinomial random variable for each time bin that contains at least one event, and the support of this multinomial is always a fixed size, $NB + 1$. When the rate of events is high, $NB + 1 \ll M$, allowing for dramatic improvements in efficiency in the discrete case.

Figure 4.1 shows the time per full Gibbs sweep as a function of the number of events per discrete time bin for the discrete and continuous formulations. The discrete formulation incurs a constant penalty whereas the continuous formulation quickly grows with the event rate. For low rates, the continuous formulation can be advantageous, but the discrete model is vastly superior in many realistic settings. For example, in Chapter 3 we worked with trades on the S&P100, which occur tens or

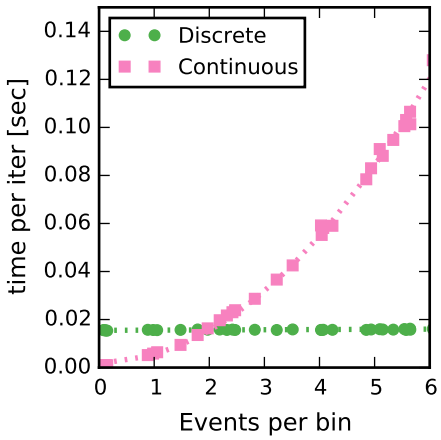

Figure 4.1: Comparison of run time per Gibbs sweep for the discrete and continuous network Hawkes formulations. Best fit lines added.

hundreds of times per second for each stock. Since the complexity of our continuous time algorithm grew with the number of events, we had to downsample the data to consider only the times when stock prices changed significantly. However, we were also looking for interactions on time scales of one minute, very large compared to the rate of trades. Thus, it is reasonable to consider a discrete time model in which the number of trades is counted in, say, 1sec bins instead. The discrete time methods of this chapter would allow us to work directly with this type of trade-level activity and still scale to days or weeks of data.

4.3 Stochastic Variational Inference

The discrete time formulation offers advantageous complexity compared to the continuous analogue, but in order to maintain the invariance of the posterior distribution, we must still work with the entire set of parents each iteration. In many cases, a subset, or "mini-batch," of time bins can provide substantial information about the global parameters of the model, and rapid progress can be made by iterating quickly over subsets of the data. This motivates our derivation of a stochastic variational inference algorithm (Hoffman et al., 2013) for this discrete time model.

Variational methods optimize a lower bound on the marginal likelihood by minimizing the KL-divergence between a tractable approximating distribution and the true posterior. Since the local parents variables, $\boldsymbol{\omega}$, are conditionally independent given the global parameters ($\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{\theta}$, etc.),

our variational approach will easily extend to the stochastic setting in which we compute unbiased estimates of the gradient of the variational objective using mini-batches of data.

The primary impediment to deriving a variational approximation is the non-conjugacy of the spike-and-slab prior on the weights. To overcome this, we approximate the spike-and-slab prior with a mixture of gamma distributions, as has previously explored by Grabska-Barwinska et al. (2013):

$$
\begin{aligned}
p(\boldsymbol{A}, \boldsymbol{W} \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) &= \prod_{n,n'} p(a_{n\to n'} \mid \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta})\, p(w_{n\to n'} \mid a_{n\to n'}, \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta}) \\
p(a_{n\to n'} \mid \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta}) &= \mathrm{Bern}(a_{n\to n'} \mid \rho_{n\to n'}), \\
p(w_{n\to n'} \mid a_{n\to n'}, \boldsymbol{z}_n, \boldsymbol{z}_{n'}, \boldsymbol{\vartheta}) &= \begin{cases} \mathrm{Gamma}(w_{n\to n'} \mid \kappa, \nu_{n\to n'}, a_{n\to n'} = 1), \\ \mathrm{Gamma}(w_{n\to n'} \mid \kappa_0, \nu_0, a_{n\to n'} = 0), \end{cases}
\end{aligned}
$$

where, as before, $\rho_{n\to n'}$ and $\nu_{n\to n'}$ are functions of the latent variables, $\boldsymbol{z}_n$ and $\boldsymbol{z}_{n'}$, and the parameters $\boldsymbol{\vartheta}$. We have approximated the "spike" in the spike-and-slab model with a gamma distribution parameterized by κ_0 and ν_0. As $\kappa_0 \to 0$ and $\nu_0 \to \infty$, the gamma distribution approaches a spike at zero.

This approximate probabilistic model is now amenable to mean field variational inference. We use a fully-factorized variational approximation, with the exception of a joint factor for each connection, $(a_{n\to n'}, w_{n\to n'})$.

$$
\begin{aligned}
q(a_{n\to n'}) &= \mathrm{Bern}(a_{n\to n'} \mid \widetilde{p}_{n\to n'}), \\
q(w_{n\to n'} \mid a_{n\to n'}) &= \begin{cases} \mathrm{Gamma}(w_{n\to n'} \mid \widetilde{\kappa}_1^{(n,n')}, \widetilde{\nu}_1^{(n,n')}, a_{n\to n'} = 1), \\ \mathrm{Gamma}(w_{n\to n'} \mid \widetilde{\kappa}_0^{(n,n')}, \widetilde{\nu}_0^{(n,n')}, a_{n\to n'} = 0)\,. \end{cases}
\end{aligned}
$$

Since the model is fully conjugate, the factors are easily derived.

Variational updates for parent variables, $q(\boldsymbol{\omega}_{t,n'})$ For the parent variables, the variational updates are

$$
\begin{aligned}
q(\boldsymbol{\omega}_{t,n'}) &= \mathrm{Mult}(\boldsymbol{\omega}_{t,n'} \mid s_{t,n'}, \widetilde{\boldsymbol{u}}_{t,n'}), \\
\widetilde{u}_{t,n'}^{(0)} &= \frac{1}{Z} \exp\left\{\mathbb{E}_{\boldsymbol{\lambda}}[\ln \lambda_{n'}^{(0)}]\right\}, \\
\widetilde{u}_{t,n'}^{(n,b)} &= \frac{1}{Z} \hat{s}_{t,n,b} \exp\left\{\mathbb{E}_{\boldsymbol{\theta}}[\ln \theta_{n\to n'}^{(b)}] + \mathbb{E}_W[\ln w_{n\to n'}]\right\},
\end{aligned}
$$

where Z is the normalization constant.

VARIATIONAL UPDATES FOR BACKGROUND RATES, $q(\lambda_n^{(0)})$ The variational form parameters of the gamma distribution over background rates are

$$\begin{aligned} q(\lambda_n^{(0)}) &= \mathrm{Gamma}(\lambda_n^{(0)} \mid \widetilde{\alpha}_\lambda^{(n)}, \widetilde{\beta}_\lambda^{(n)}), \\ \widetilde{\alpha}_\lambda^{(n)} &= \alpha_\lambda + \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\omega}} \left[\omega_{t,n}^{(0)} \right], \\ \widetilde{\beta}_\lambda^{(n)} &= \beta_\lambda + T\Delta t \,. \end{aligned}$$

VARIATIONAL APPROXIMATION FOR IMPULSE RESPONSE PARAMETERS, $q(\boldsymbol{\theta}_{n\to n'})$ With the conjugate prior formulation the variational parameter updates for the Dirichlet distributed impulse response parameters are

$$\begin{aligned} q(\boldsymbol{\theta}_{n\to n'}) &= \mathrm{Dir}(\boldsymbol{n}_{n\to n'} \mid \widetilde{\boldsymbol{\gamma}}^{(n,n')}), \\ \widetilde{\gamma}_b^{(n,n')} &= \gamma_b + \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\omega}} \left[\omega_{t,n'}^{(n,b)} \right] . \end{aligned}$$

VARIATIONAL APPROXIMATION FOR THE WEIGHTED ADJACENCY MATRIX. The primary motivation for adopting a weakly sparse mixture of gamma distributions is to derive an efficient variational inference algorithm. The mixture-of-gammas prior is conjugate with the Poisson observations, and hence the variational distribution is also a mixture of gammas:

$$\begin{aligned} q(w_{n\to n'} \mid a_{n\to n'} = 1) &= \mathrm{Gamma}(w_{n\to n'} \mid \widetilde{\kappa}_1^{(n,n')}, \widetilde{\nu}_1^{(n,n')}) \\ \widetilde{\kappa}_1^{(k,k')} &= \kappa + \sum_{t=1}^{T} \sum_{b=1}^{B} \mathbb{E}_{\boldsymbol{\omega}} \left[\omega_{t,n'}^{(n,b)} \right] \\ \widetilde{\nu}_1^{(n,n')} &= \mathbb{E}_\nu[\nu_{n\to n'}] + \sum_{t=1}^{T} s_{t,n} \,, \end{aligned}$$

and likewise for the "spike" factor,

$$
\begin{aligned}
q(w_{n\to n'} \mid a_{n\to n'} = 0) &= \mathrm{Gamma}(w_{n\to n'} \mid \widetilde{\kappa}_0^{(n,n')}, \widetilde{\nu}_0^{(n,n')}) \\
\widetilde{\kappa}_0^{(k,k')} &= \kappa_0 + \sum_{t=1}^{T} \sum_{b=1}^{B} \mathbb{E}_{\boldsymbol{\omega}} \left[\omega_{t,n'}^{(n,b)} \right] \\
\widetilde{\nu}_0^{(n,n')} &= \nu_0 + \sum_{t=1}^{T} s_{t,n} \,.
\end{aligned}
$$

This leaves us with $q(a_{n\to n'})$, which is Bernoulli distributed with parameter $\widetilde{p}_{n\to n'}$. The optimal parameter is given by,

$$
\begin{aligned}
\frac{\widetilde{p}_{n\to n'}}{1-\widetilde{p}_{n\to n'}} = \frac{\exp\{\mathbb{E}[\ln \rho_{n\to n'}]\}}{\exp\{\mathbb{E}[\ln(1-\rho_{n\to n'})]\}} \times \\
\frac{(\exp\{\mathbb{E}[\ln \nu_{n\to n'}]\})^{\kappa}}{\Gamma(\kappa)} \times \frac{\Gamma(\widetilde{\kappa}_1^{(n,n')})}{(\widetilde{\nu}_1^{(n,n')})^{\widetilde{\kappa}_1^{(n,n')}}} \times \frac{\Gamma(\kappa_0)}{(\nu_0)^{\kappa_0}} \times \frac{(\widetilde{\nu}_0^{(n,n')})^{\widetilde{\kappa}_0^{(n,n')}}}{\Gamma(\widetilde{\kappa}_0^{(n,n')})} \,.
\end{aligned}
$$

As with Gibbs sampling, we assume a variational approximation for the network model can be derived, and provide access to the necessary expectations, $\mathbb{E}[\ln \rho_{n\to n'}]$, $\mathbb{E}[\ln(1-\rho_{n\to n'})]$, $\mathbb{E}[\nu_{n\to n'}]$ and $\mathbb{E}[\ln \nu_{n\to n'}]$.

The spike counts in each time bin are conditionally independent given the network weights and the adjacency matrix — a common pattern exploited by stochastic variational inference (SVI) algorithms (Hoffman et al., 2013). These methods optimize the variational objective using stochastic gradient methods that work with mini-batches of data. Often, a mini-batch of data can provide valuable information about the global parameters, in our case the network and background rates. Quickly iterating over these global parameters allows us to reach good modes of the posterior distribution in a fraction of the time that standard variational Bayes and Gibbs sampling require, since those methods must process the entire dataset before making an update. SVI does require some tuning, however. In particular, we must set a mini-batch size and a step size schedule. In this work, we fix the mini-batch size to $T_{\mathsf{mb}} = 1024$ and set the step size at iteration i to $(i+1)^{-0.5}$. These parameters may be tuned with general purpose hyperparameter optimization techniques (Snoek et al., 2012).

(a) Short dataset: $T = 10^4$ **(b)** Long dataset: $T = 10^5$

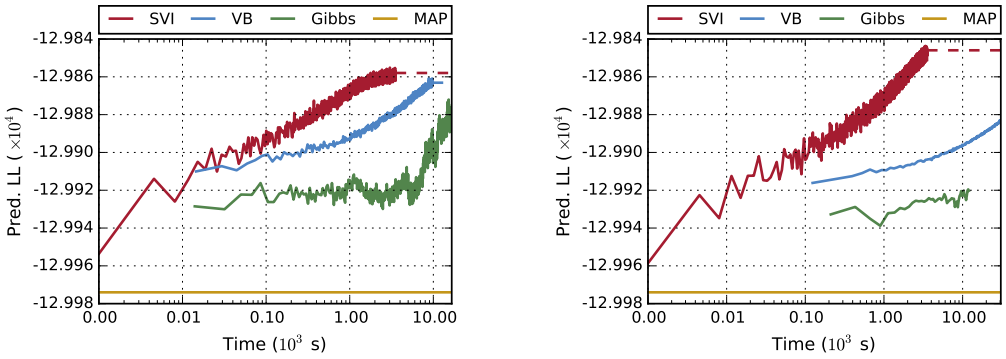

Figure 4.2: Predictive log likelihood versus wall clock time for three Bayesian inference algorithms on a dataset of $N = 50$ neurons and $T = 10^4$ and $T = 10^5$ time bins on the left and right, respectively.

4.4 Synthetic Results

We assess the performance of the proposed inference algorithms on a synthetic dataset generated by a strongly sparse Hawkes process with $N = 50$ neurons. We used a stochastic block model network prior with $K = 5$ clusters, each consisting of ten densely connected processes ($p_{k \to k} = 0.4$), with sparse connections to processes in other clusters ($p_{k \to k'} = 0.01$). All weights share the same scale of $\nu = 5.0$, though this information is not provided *a priori*. We simulate $T = 10^5$ time bins in steps of size $\Delta t = 1$. The neurons have an mean background rate of 1.0 event per time bin and, due to the network interactions, the average total rate of the processes is 16.7 ± 12.0 events per bin. Referring to Figure 4.1, this is a regime that favors the discrete model. We initialized by performing MAP estimation on the first $T_{\text{init}} = 10^4$. Then we trained the model using Gibbs sampling, batch variational Bayesian inference, and stochastic variational inference,

We trained the models on only the first 10^4 time bins, the same that were used for initialization. We evaluated the algorithms in terms of their predictive log likelihood on a held-out dataset of length $T_{\text{test}} = 10^3$. Figure 4.2a shows the results as a function of wall-clock time. We find that SVI obtains competitive predictive log likelihood in a matter of minutes. Batch VB and Gibbs converge at a considerably slower rate, though they eventually match the SVI predictive likelihood after hours of computation. The MAP estimate, even with cross validated regularization, underperforms the other competing algorithms.

This trend is exaggerated when we consider the entire training set of size $T = 10^5$. Figure 4.2b il-

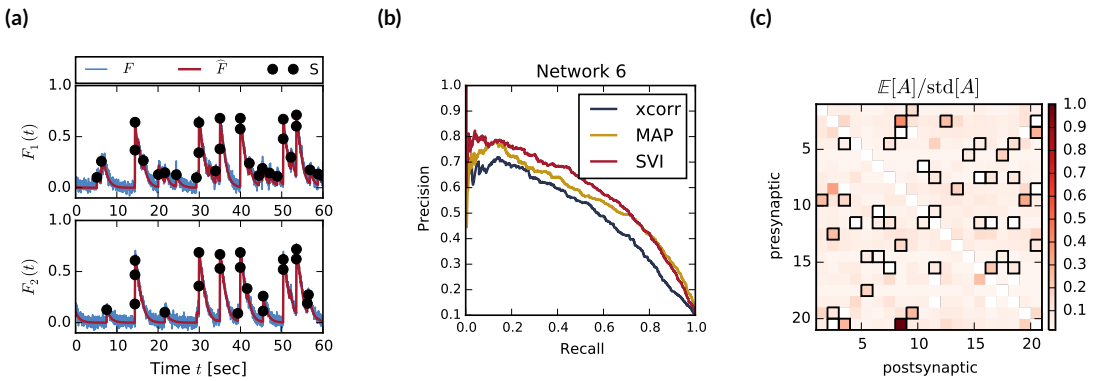

Figure 4.3: Application of the network Hawkes model to a connectomics challenge. **(a)** The data is in the form of a calcium fluorescence trace, which we preprocess to extract neural spike times. **(b)** We measure performance on a link prediction task using a precision-recall curve and find that the posterior estimates of SVI provide the best estimates on some networks. In addition to an estimate of the connection probability and weight, SVI provides an estimate of the posterior uncertainty. **(c)** Inferred $\mathbb{E}_q[\boldsymbol{A}]/\mathrm{std}_q[\boldsymbol{A}]$ for the first 20 neurons. True connections are outlined in black.

lustrates the power of SVI in handling these large time datasets. Considerable information about the global parameters (e.g., the network) can be gained from just a mini-batch of time points. Hence, we can make rapid improvements in predictive log likelihood very quickly. By contrast, each step of the Gibbs and batch VB algorithms is approximately 10 times slower, and even after computing sufficient statistics over the entire dataset, the algorithm is only able to make limited progress per iteration.

4.5 Connectomics Results

We tested these inference algorithms on the data from the Chalearn neural connectomics challenge* (Stetter et al., 2012). The data consist of calcium fluorescence traces, $\boldsymbol{F}$, from six networks of $N = 100$ neurons each. We use ten minutes of data at 50Hz sampling frequency to yield $T = 3 \times 10^6$ entries in $\boldsymbol{S}$. In this case, the networks are purely excitatory, and each action potential, or spike, increases the probability of the downstream neurons firing as a result. This matches the underlying intuition of the Hawkes process model, making it a natural choice.

In order to apply the Hawkes model, we first convert the fluorescence traces into a spike count

*http://connectomics.chalearn.org

	Network 1		Network 2		Network 3		Network 4		Network 5	
Algorithm	ROC	PRC	ROC	PRC	ROC	PRC	ROC	PRC	ROC	PRC
xcorr	0.596	0.139	0.591	0.133	**0.701**	**0.198**	0.745	0.296	0.798	0.359
MAP	0.607	0.174	**0.619**	**0.143**	0.698	0.178	**0.790**	0.334	**0.859**	0.408
SVI	**0.649**	**0.184**	0.605	0.141	0.673	0.176	0.774	**0.342**	0.844	**0.410**

Table 4.1: Comparison of inference algorithms on link prediction for five networks from the Chalearn connectomics challenge. Performance is measured by area under the ROC curve and area under the precision recall curve (PRC). In four of the five networks a Hawkes process model provides the best results.

matrix using OOPSI, a Bayesian inference algorithm based on a model of calcium fluorescence (Vogelstein et al., 2010). The output is a filtered fluorescence trace, $\widehat{\boldsymbol{F}}$, and a probability of spike for each time bin. We threshold this at probability 0.7 to get a $T \times N$ binary spike matrix, $\boldsymbol{S}$. This preprocessing is shown in Figure 4.3a.

Figure 4.3b shows the precision-recall curve we used to evaluate the algorithms' performance on network recovery. As a baseline, we compare against simple thresholding of the cross correlation matrix. On Network 6, SVI offers the best network inference. Table 4.1 shows the results on the other five networks using the same model parameters. On 4/5 of these networks, the Bayesian methods offer the best performance.

Figure 4.3c illustrates one of the main advantages of the fully Bayesian inference algorithm – calibrated estimates of posterior uncertainty. Here we show the SVI algorithm's estimate of the posterior mean of $\boldsymbol{A}$ normalized by the posterior standard deviation for a subset of 20 neurons from Network 6. We also outline the true connections to show that the most confident predictions are more likely to correspond to true connections. Such estimates of the posterior uncertainty are not available with standard heuristic methods or point estimates.

4.6 Conclusion

This brief chapter provided a link between the ideas introduced in Chapter 3 — namely the combination of network models and point process observations — to the discrete time autoregressive models of the next few chapters. We also showed how the conditional independence of the spike counts could be leveraged in a stochastic variational inference algorithm that scales to long recording durations. The key, again, was the Poisson superposition principle, which allowed a simple auxiliary variable formulation. Combining this formulation with an approximate spike-and-slab model led to a fully-conjugate model that admitted an efficient inference algorithm.

In the next chapter, we will continue to build on these ideas, but we will address a major limita-

tion of this approach. The Poisson superposition principle only applies to *linear* models. Since the rate must be nonnegative, linear models cannot have inhibitory interactions with negative weights. We will show how this limitation can be overcome with another clever auxiliary variable trick.

5

Networks with Nonlinear Autoregressive Dynamics

The last two chapters combined network models and Hawkes processes to construct probabilistic models for dynamic neural spike trains. The key assumption of Hawkes processes, the assumption we leveraged to derive efficient inference algorithms, was that the firing rate was the sum of non-negative impulse responses from preceding spikes. In other words, the interactions in Hawkes processes were additive and purely excitatory. While this led to interpretable network models for some systems, in many cases it is more natural to expect a mix of excitatory and inhibitory interactions. Unfortunately, we cannot have inhibitory interactions within a simple additive model because they could yield negative firing rates. Instead, we need to revisit the assumption of linear dynamics.

As we discussed in Chapter 2, there is a simple way to generalize the linear autoregressive dynamics. We retain the linear combination of impulse responses from preceding events, but then we apply a rectifying nonlinear function to obtain a firing rate. Formally, we assume the following model:

$$\begin{aligned}\psi_{t,n} &= \psi_n^{(0)} + \sum_{n'=1}^{N}\sum_{d=1}^{D} s_{t-d,n'} \cdot h_{n'\to n}[d], \\ \lambda_{t,n} &= g(\psi_{t,n}),\end{aligned}$$

where $\psi_{t,n}$ is a real-valued signal called the *activation*, and $g : \mathbb{R} \to \mathbb{R}_+$ is a rectifying nonlinear function that converts the activation into a firing rate. Once this nonlinearity has been introduced, the impulse response functions, $h_{n'\to n}[d]$, are free to be both positive and negative.

As before, we assume the spike count is randomly drawn from a discrete distribution parameterized by $\lambda_{t,n}$. In this chapter, however, we consider a more general class of observation distributions with neuron-specific global parameters ν_n,

$$s_{t,n} \sim p(s_{t,n} \mid \lambda_{t,n}, \nu_n).$$

This nonlinear autoregressive model is also known as a generalized linear model (GLM), and it is widely used in neuroscience (Paninski, 2004; Truccolo et al., 2005). For example, these models have proven useful in elucidating correlated patterns of activity in simultaneously recorded populations of retinal ganglion cells (RGCs) (Pillow et al., 2008). By elaborating upon the basic generalized linear model, more sophisticated structure of the underlying bipolar cells has been revealed from RGC activity (Freeman et al., 2015). While the weights of the GLM cannot typically be interpreted as synaptic connections (Vidne et al., 2012), these models have nevertheless had some success extracting underlying synaptic connectivity from spike trains (Gerhard et al., 2013; Fletcher et al., 2011; Soudry et al., 2015). As we move toward larger and more complete recordings of neural circuits, generalized linear models are likely to play a major role in guiding our understanding of neural systems.

If the spike counts are Poisson distributed, then under general conditions (Paninski, 2004) the likelihood of the spike counts will be log concave and amenable to efficient, exact *maximum a posteriori* (MAP) estimation. Moreover, fully Bayesian estimation procedures based on expectation-propagation (Gerwinn et al., 2008) and Hamiltonian Monte Carlo (Ahmadian et al., 2011) have given some insight into the posterior distribution of network weights. Unfortunately, once we introduce nontrivial prior distributions, like network models, the posterior becomes multimodal and Bayesian inference becomes more challenging. However, see Soudry et al. (2015) for some recent advances in approximate MAP estimation in models that combine network priors and generalized linear models of spike trains.

This chapter develops efficient MCMC algorithms for approximating the posterior distribution of weights in a GLM with prior distributions on the impulse responses that are derived from probabilistic network models. Unfortunately, the augmentation scheme developed for linear Hawkes processes is no longer viable due to the nonlinear interactions. Instead, we develop an inference algorithm based on a recently developed scheme known as Pólya-gamma augmentation (Polson et al., 2013). A recent application of these methods to factor analysis models for neural spike trains provided the motivation for the work in this chapter (Pillow and Scott, 2012). The basic idea behind the Pólya-gamma augmentation is to introduce a set of auxiliary variables conditioned upon which the

Name	Parameters	$p(s \mid \psi, \nu)$	$\mathbb{E}[s]$	$\mathrm{Var}(s)$
Gaussian	$\mathcal{N}(\psi, \nu)$	$\frac{1}{\sqrt{2\pi\nu}} e^{-\frac{1}{2\nu}(s-\psi)^2}$	ψ	ν
Bernoulli	$\mathrm{Bern}(\sigma(\psi))$	$\frac{(e^{\psi})^s}{1+e^{\psi}}$	$\sigma(\psi)$	$\sigma(\psi)\,\sigma(-\psi)$
Binomial	$\mathrm{Bin}(\nu, \sigma(\psi))$	$\binom{\nu}{s} \frac{(e^{\psi})^s}{(1+e^{\psi})^{\nu}}$	$\nu\sigma(\psi)$	$\nu\sigma(\psi)\,\sigma(-\psi)$
Neg. Binomial	$\mathrm{NB}(\nu, \sigma(\psi))$	$\binom{\nu+s-1}{s} \frac{(e^{\psi})^s}{(1+e^{\psi})^{\nu+s}}$	νe^{ψ}	$\nu e^{\psi}/\sigma(-\psi)$

Table 5.1: List of observation distributions.

discrete spike counts actually "look" like Gaussian observations. Once we have reduced the problem to inference in a linear Gaussian model, a host of efficient inference algorithms are at our disposal.

5.1 Probabilistic Model

The underlying model is very similar to the linear autoregressive models of the preceding chapter. Rather than considering Poisson distributed spike counts, however, here we work with other discrete observation models that are amenable to Pólya-gamma augmentation. As we generalize to allow both excitatory and inhibitory interactions, we revisit the form of our impulse response and introduce a simpler form. Finally, we outline a variety of network models that provide multivariate Gaussian prior distributions on the interaction weights.

5.1.1 Spike count models

The spike counts are stochastically drawn from a distribution parameterized by an underlying, real-valued *activation*, $\psi_{t,n}$, and a static parameter, ν_n. The Pólya-gamma augmentation can be applied to observation models that depend on a logistic transformation of the activation,

$$\sigma(\psi) = \frac{e^{\psi}}{1 + e^{\psi}},$$

which has the property $\sigma(-\psi) = 1 - \sigma(\psi)$. The Bernoulli, binomial, and negative binomial distributions are all natural choices. While the Gaussian distribution is not a proper model for discrete spike counts, we include it in our list since our inference procedure will reduce the other models to the Gaussian case. Table 5.1 lists the basic properties of these observation distributions.

The Bernoulli distribution is appropriate for binary spike counts, whereas the binomial and negative binomial have support for integer $s \in \{0, \ldots, \nu\}$ and $s \in \{0, 1, \ldots\}$, respectively. Notably

missing from this list is the Poisson distribution, which is not amenable to the Pólya-gamma augmentation. Nevertheless, both the binomial and negative binomial distributions converge to the Poisson under certain limits. For example, $\lim_{r\to\infty} \mathrm{NB}(r, \sigma(\psi - \log r)) = \mathrm{Poisson}(e^{\psi})$. Moreover, the binomial and negative binomial distributions afford the added flexibility of modeling under- and over-dispersed spike counts. Specifically, while the Poisson has unit dispersion (its mean is equal to its variance), the binomial distribution is always under-dispersed, with variance less than its mean, and the negative binomial is always over-dispersed, with variance greater than its mean.

5.1.2 Linear Gaussian Activation Model

As in the Chapters 3 and 4, we model the impulse response as a weighted sum of basis functions,

$$h_{n'\to n}[d] = a_{n'\to n} \sum_{b=1}^{B} w_{n'\to n}^{(b)} \cdot \phi_b[d]. \tag{5.1}$$

We dispense with the normalized basis function, $\hbar[d]$, because, in the nonlinear model, the normalized basis functions can no longer be seen as distributions over child spike times. Indeed the notion of "parent" and "child" spikes is no longer warranted since the Poisson superposition principle does not apply to nonlinear models. Moreover, in some cases, we may wish to model interactions that are both excitatory *and* inhibitory — for example, the effect of a spike may be inhibitory at short time scales and excitatory after a delay. While these types of interactions may not correspond to actual biological synapses, they can still capture salient correlations in neural spike trains.

Plugging Eq. 5.1 into the activation model, we have,

$$\begin{aligned}
\psi_{t,n} &\triangleq \psi_n^{(0)} + \sum_{n'=0}^{N} \sum_{b=1}^{B} a_{n'\to n}\, w_{n'\to n}^{(b)} \left(\sum_{d=1}^{D} \phi_b[d] \cdot s_{t-d,n'} \right) \\
&= \psi_n^{(0)} + \sum_{n'=0}^{N} \sum_{b=1}^{B} a_{n'\to n}\, w_{n'\to n}^{(b)}\, \widehat{s}_{t,n'}^{(b)} \\
&= (\boldsymbol{a}_n \odot \boldsymbol{w}_n)^{\mathsf{T}}\, \widehat{\boldsymbol{s}}_t,
\end{aligned} \tag{5.2}$$

where $\psi_n^{(0)}$ is the baseline activation of neuron n, $\boldsymbol{\phi}_b$ is a basis function that weights the previous spike counts for offsets $d \in \{1, \ldots, D\}$, the binary variable $a_{n'\to n} \in \{0, 1\}$ indicates whether or not there exists a directed connection from neuron n' to neuron n, and $w_{n'\to n}^{(b)}$ captures the influence that spikes on neuron n' exert on neuron n at offsets weighted by the b-th basis function. Since

the basis function and the signal are assumed to be fixed, we precompute the inner sum, which is simply the convolution of the signal with the basis function, to get $\widehat{s}_{t,n}^{(b)}$. Since this is a linear function, we combine the connections, weights, and filtered spike trains into vectors to get the linear form in (5.2). Here, we have,

$$\begin{aligned}
\boldsymbol{a}_n &= \Big[1, \quad a_{1\to n}, \quad \ldots, \quad a_{1\to n}, \quad \ldots, \quad a_{N\to n}, \quad \ldots, \quad a_{N\to n}\Big]^\mathsf{T}, \\
\boldsymbol{w}_n &= \Big[\psi_n^{(0)}, \quad w_{1\to n}^{(1)}, \quad \ldots, \quad w_{1\to n}^{(B)}, \quad \ldots, \quad w_{N\to n}^{(1)}, \quad \ldots, \quad w_{N\to n}^{(B)}\Big]^\mathsf{T}, \\
\widehat{\boldsymbol{s}}_t &= \Big[1, \quad \widehat{s}_{t,1}^{(1)}, \quad \ldots, \quad \widehat{s}_{t,1}^{(B)}, \quad \ldots, \quad \widehat{s}_{t,N}^{(1)}, \quad \ldots, \quad \widehat{s}_{t,N}^{(B)}\Big]^\mathsf{T},
\end{aligned}$$

and $\odot$ denotes the elementwise product. Note that each $a_{n'\to n}$ is repeated B times in the vector $\boldsymbol{a}_n$. For convenience, we let $\boldsymbol{A}$ and $\boldsymbol{W}$ refer to the $N \times NB + 1$ matrices obtained by stacking the vectors $\boldsymbol{a}_n^\mathsf{T}$ and $\boldsymbol{w}_n^\mathsf{T}$, and we let $\widehat{\boldsymbol{S}}$ denote the $T \times NB + 1$ matrix with rows given by $\widehat{\boldsymbol{s}}_t^\mathsf{T}$. The major difference between this formulation and that of the standard GLM is that here we have explicitly modeled the sparsity of the weights via the "adjacency matrix" $\boldsymbol{A}$. Under the standard formulation, all weights are present, that is, $a_{n'\to n} \equiv 1$.

Consider a model with one basis function ($B = 1$) defined by, $\phi_{1,d} = e^{-d/\tau}$. Then $\widehat{s}_{t,n}^{(1)}$ is a weighted sum of spikes in the window $[t - D, t - 1]$, where the weights decay according to an exponential function with time constant τ. If $a_{n'\to n} = 1$, indicating a connection from neuron n' to neuron n, and the weight, $w_{n'\to n}^{(1)}$, is positive, the influence will be excitatory. If it is negative, the effect will be inhibitory. Together, the weights $\boldsymbol{W}$ define a functional *network* of interactions.

5.1.3 Network Models

With this new impulse response model, each edge of the network is now associated with a B-dimensional weight vector. We can use the same adjacency matrix models as in the previous chapters, but we need to consider new models for the weight matrix. A multivariate Gaussian prior is a natural choice. We consider weight models of the form,

$$\begin{aligned}
p(\boldsymbol{w}_n \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) &= \mathcal{N}(\psi_n^{(0)} \mid \mu_0, \sigma_0^2) \prod_{m=1}^{N} \mathcal{N}(\boldsymbol{w}_{n'\to n} \mid \boldsymbol{\mu}_{n'\to n}, \boldsymbol{\Sigma}_{n'\to n}), \\
&= \mathcal{N}(\boldsymbol{w}_n \mid \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n),
\end{aligned}$$

Name	$\mathrm{dom}(\boldsymbol{z}_n)$	$\boldsymbol{\mu}_{n' \to n}$	$\boldsymbol{\Sigma}_{n' \to n}$
Gaussian Model	—	$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$
Stochastic Block Model	$\{1, \ldots, K\}$	$\boldsymbol{\mu}_{z_{n'} \to z_n}$	$\boldsymbol{\Sigma}_{z_{n'} \to z_n}$
Latent Distance Model ($B = 1$)	$\mathbb{R}^K$	$-\lvert\lvert \boldsymbol{z}_n - \boldsymbol{z}_{n'} \rvert\rvert_2^2 + \gamma_0$	σ^2

Table 5.2: Gaussian Weight Models

where $\boldsymbol{\mu}_{n' \to n}$ and $\boldsymbol{\Sigma}_{n' \to n}$ are the mean and covariance, and they implicitly depend on the latent variables, $\boldsymbol{z}_{n'}$ and $\boldsymbol{z}_n$, and the parameters of the network model, $\boldsymbol{\vartheta}$. The last line combines the Gaussian factors into single multivariate Gaussian prior with parameters,

$$\boldsymbol{\mu}_n = \begin{bmatrix} \mu_0 \\ \boldsymbol{\mu}_{1 \to n} \\ \vdots \\ \boldsymbol{\mu}_{N \to n} \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{\Sigma}_n = \begin{bmatrix} \sigma_0^2 & & & \\ & \boldsymbol{\Sigma}_{1 \to n} & & \\ & & \ddots & \\ & & & \boldsymbol{\Sigma}_{N \to n} \end{bmatrix}. \tag{5.3}$$

Table 5.2 defines the three weight models considered in this chapter. Each model defines the mean and variance of a multivariate normal distribution, $\boldsymbol{w}_{n' \to n} \sim \mathcal{N}(\boldsymbol{\mu}_{n' \to n}, \boldsymbol{\Sigma}_{n' \to n})$. In the Gaussian model, all weights are independent and identically distributed. The stochastic block model, has parameters for each pair of classes, each drawn from a normal inverse-Wishart prior. Finally, we consider a latent distance model, but only for the case where the weights are scalar, i.e. $B = 1$. In this case, the distance between points is inversely proportional to the mean weight. For higher order weights, additional assumptions would be required in order to relate distance to vector weights. In this model, we assume standard normal priors on the parameters $\boldsymbol{z}_n$ and γ_0. The variance is given an inverse gamma prior, $\sigma^2 \sim \mathrm{IGa}(\alpha, \beta)$.

5.2 Inference via Gibbs Sampling

Inference is the process of evaluating the posterior distribution over latent variables given the observed signal, which is related to the joint distribution by Bayes' rule:

$$p(\{\nu_n, \boldsymbol{a}_n, \boldsymbol{w}_n, \boldsymbol{z}_n\}_{n=1}^N, \boldsymbol{\vartheta} \mid \boldsymbol{S}) = \frac{p(\boldsymbol{S}, \{\nu_n, \boldsymbol{a}_n, \boldsymbol{w}_n, \boldsymbol{z}_n\}_{n=1}^N, \boldsymbol{\vartheta})}{p(\boldsymbol{S})}.$$

It is computationally intractable to compute this posterior exactly and it has no simple closed form solution, so we must instead resort to approximate methods. We use Markov chain Monte Carlo (MCMC) methods to collect samples from this posterior distribution.

5.2.1 Collapsed Gibbs Network Updates

The most challenging aspect of inference is sampling the posterior distribution over connections, $\boldsymbol{A}$. In the dense model, where $a_{n' \to n} \equiv 1$, the posterior distribution over weights is often log concave, which makes it easy to find the *maximum a posteriori* (MAP) estimate and characterize the local uncertainty around the most likely weights. When the connectivity matrix is sparse, there are instead many modes corresponding to different patterns of connectivity. While this makes inference more challenging, sparse connectivity is an important feature that contributes to the interpretability of the model.

Fortunately, we can make posterior inference of the network considerably more efficient by integrating over possible weights and sampling the binary adjacency matrix from its marginal distribution. First, consider the Gaussian observation model. Since $\psi_{t,n}$ is linear in $\boldsymbol{w}_n$, the likelihood is conjugate with the Gaussian prior, and hence the posterior is Gaussian as well. We compute the posterior distribution in closed form:

$$\begin{aligned} p(\boldsymbol{w}_n \mid \boldsymbol{S}, \boldsymbol{a}_n, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) &\propto \mathcal{N}(\boldsymbol{w}_n \mid \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \prod_{t=1}^{T} \mathcal{N}(s_{t,n} \mid (\boldsymbol{a}_n \odot \boldsymbol{w}_n)^\mathsf{T} \widehat{\boldsymbol{s}}_t,\, \nu_n) \\ &= \mathcal{N}(\boldsymbol{w}_n \mid \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\, \mathcal{N}(\boldsymbol{s}_n \mid (\boldsymbol{a}_n \odot \boldsymbol{w}_n)^\mathsf{T} \widehat{\boldsymbol{S}},\, \nu_n \boldsymbol{I}) \\ &\propto \mathcal{N}(\boldsymbol{w}_n \mid \widetilde{\boldsymbol{\mu}}_n, \widetilde{\boldsymbol{\Sigma}}_n), \end{aligned} \tag{5.4}$$

where

$$\begin{aligned} \widetilde{\boldsymbol{\Sigma}}_n &= \left[\boldsymbol{\Sigma}_n^{-1} + \left(\widehat{\boldsymbol{S}}^\mathsf{T} (\nu_n^{-1} \boldsymbol{I}) \widehat{\boldsymbol{S}}\right) \odot (\boldsymbol{a}_n \boldsymbol{a}_n^\mathsf{T})\right]^{-1}, \\ \widetilde{\boldsymbol{\mu}}_n &= \widetilde{\boldsymbol{\Sigma}}_n \left[\boldsymbol{\Sigma}_n^{-1} \boldsymbol{\mu}_n + \left(\widehat{\boldsymbol{S}}^\mathsf{T} (\nu_n^{-1} \boldsymbol{I}) \boldsymbol{s}_n\right) \odot \boldsymbol{a}_n\right]. \end{aligned}$$

Given this closed-form Gaussian conditional, we can also compute the conditional distribution

over just $\boldsymbol{a}_n$, integrating out the corresponding weights, $\boldsymbol{w}_n$:

$$\begin{aligned} p(\boldsymbol{a}_n \mid \widehat{\boldsymbol{S}}, \boldsymbol{\rho}_n, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) &= \int p(\boldsymbol{a}_n, \boldsymbol{w}_n \mid \boldsymbol{S}, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) \, \mathrm{d}\boldsymbol{w}_n \\ &\propto p(\boldsymbol{a}_n \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) \int p(\boldsymbol{w}_n \mid \boldsymbol{S}, \boldsymbol{a}_n, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) \, \mathrm{d}\boldsymbol{w}_n \\ &= p(\boldsymbol{a}_n \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}) \, \frac{\left|\boldsymbol{\Sigma}_n\right|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \boldsymbol{\mu}_n^\mathsf{T} \boldsymbol{\Sigma}_n^{-1} \boldsymbol{\mu}_n \right\}}{\left|\widetilde{\boldsymbol{\Sigma}}_n\right|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \widetilde{\boldsymbol{\mu}}_n^\mathsf{T} \widetilde{\boldsymbol{\Sigma}}_n^{-1} \widetilde{\boldsymbol{\mu}}_n \right\}}. \end{aligned} \tag{5.5}$$

Thus, we can efficiently sample from the conditional distribution of $\boldsymbol{a}_n$ and $\boldsymbol{w}_n$ by first iterating over each neuron $n' \in \{1, \ldots, N\}$ and sampling a new value of $a_{n' \to n}$, fixing the values of $a_{n'' \to n}$ for $n'' \neq n'$ and integrating out the value of $\boldsymbol{w}_n$. To do so, we simply evaluate the marginal probability in Eq. 5.5 for both values of $a_{n' \to n}$ and sample accordingly. Moreover, note that $\widetilde{\boldsymbol{\Sigma}}_n$ reduces to the prior where $a_{n' \to n} = 0$. This will lead to $B \times B$ diagonal blocks that are equal in both the numerator and the denominator. Thus, the terms for which $a_{n' \to n} = 0$ will cancel in the log determinant and quadratic form of 5.5. As a result, if $\boldsymbol{A}$ is p-sparse (i.e. each neuron has at most p incoming edges) evaluating the marginal probability of $\boldsymbol{a}_n$ has complexity an $O(p^3)$. Once $\boldsymbol{a}_n$ has been completely resampled, we can sample a new value of $\boldsymbol{w}_n$ from its multivariate Gaussian conditional distribution, given by Eq. 5.4.

5.2.2 Pólya-gamma augmentation for discrete observations

When the observations are not Gaussian, the conditional distribution of $\boldsymbol{w}_n$ cannot be computed in closed form and the collapsed updates are intractable. To circumvent this problem, we leverage recently developed augmentation schemes for Gaussian models with discrete observations (Polson et al., 2013; Pillow and Scott, 2012). The idea is to augment the observations, $s_{t,n}$, with auxiliary variables, $\omega_{t,n}$, such that conditioned upon the auxiliary variables, the discrete likelihood appears Gaussian.

First, notice that the discrete likelihoods in Table 5.1 can all be put into a "standard" form in which the probability mass function can be written,

$$p(s \mid \psi, \nu) = c(s, \nu) \, \frac{(e^{\psi})^{a(s,\nu)}}{(1 + e^{\psi})^{b(s,\nu)}},$$

for some functions, a, b, and c that do not depend on ψ. The integral identity at the heart of the

Pólya-gamma augmentation scheme is

$$\frac{(e^{\psi})^a}{(1+e^{\psi})^b} = 2^{-b} e^{\kappa\psi} \int_0^\infty e^{-\omega\psi^2/2}\, p_{\mathrm{PG}}(\omega \mid b, 0)\, \mathrm{d}\omega, \tag{5.6}$$

where $\kappa = a - b/2$ and $p(\omega \mid b, 0)$ is the density of the Pólya-gamma distribution $\mathrm{PG}(b, 0)$, which does not depend on ψ.

Using Eq. 5.6 along with priors $p(\psi)$ and $p(\nu)$, we can write the joint density of (ψ, s, ν) as

$$\begin{aligned} p(s, \nu, \psi) &= p(\nu)\, p(\psi)\, c(s, \nu) \frac{(e^{\psi})^{a(s,\nu)}}{(1+e^{\psi})^{b(s,\nu)}} \\ &= \int_0^\infty p(\nu)\, p(\psi)\, c(s, \nu)\, 2^{-b(s,\nu)} e^{\kappa(s,\nu)\psi} e^{-\omega\psi^2/2}\, p_{\mathrm{PG}}(\omega \mid b(s, \nu), 0)\, \mathrm{d}\omega. \end{aligned} \tag{5.7}$$

The integrand of Eq. 5.7 defines a joint density on (s, ν, ψ, ω) which admits $p(s, \nu, \psi)$ as a marginal density. Conditioned on these auxiliary variables ω, we have that the likelihood as a function of ψ is,

$$p(s \mid \psi, \nu, \omega) \propto e^{\kappa(s,\nu)\psi} e^{-\omega\psi^2/2} \propto \mathcal{N}\left(\frac{\kappa(s, \nu)}{\omega} \,\middle|\, \psi,\ \frac{1}{\omega}\right).$$

Thus, we effectively have a Gaussian likelihood for ψ, after conditioning on s and ω. Now we can apply this augmentation scheme to the full model, introducing auxiliary variables, $\omega_{t,n}$ for each spike count, $s_{t,n}$. Given these variables, the conditional distribution of $\boldsymbol{w}_n$ can be computed in closed form, as before. Let,

$$\boldsymbol{\kappa}_n = \begin{bmatrix} \kappa(s_{1,n}, \nu_n), & \ldots, & \kappa(s_{T,n}, \nu_n) \end{bmatrix}^{\mathsf{T}},$$

and

$$\boldsymbol{\Omega}_n = \mathrm{diag}\left(\begin{bmatrix} \omega_{1,n}, & \ldots, & \omega_{T,n} \end{bmatrix}\right).$$

Then we have $p(\boldsymbol{w}_n \mid \boldsymbol{S}, \boldsymbol{a}_n, \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}, \boldsymbol{\omega}_n, \nu_n) \propto \mathcal{N}(\boldsymbol{w}_n \mid \widetilde{\boldsymbol{\mu}}_n, \widetilde{\boldsymbol{\Sigma}}_n)$, where

$$\begin{aligned} \widetilde{\boldsymbol{\Sigma}}_n &= \left[\boldsymbol{\Sigma}_n^{-1} + \left(\widehat{\boldsymbol{S}}^{\mathsf{T}} \boldsymbol{\Omega}_n \widehat{\boldsymbol{S}}\right) \odot (\boldsymbol{a}_n \boldsymbol{a}_n^{\mathsf{T}})\right]^{-1}, \\ \widetilde{\boldsymbol{\mu}}_n &= \widetilde{\boldsymbol{\Sigma}}_n \left[\boldsymbol{\Sigma}_n^{-1} \boldsymbol{\mu}_n + \left(\widehat{\boldsymbol{S}}^{\mathsf{T}} \boldsymbol{\kappa}_n\right) \odot \boldsymbol{a}_n\right]. \end{aligned}$$

Having introduced auxiliary variables, we must now also derive Markov transitions to update them as well. Fortunately, the Pólya-gamma distribution is designed such that the conditional density of the auxiliary variables is just a "tilted" Pólya-gamma density,

$$p(\omega_{t,n} \mid s_{t,n}, \nu_n, \psi_{t,n}) = p_{\mathrm{PG}}(\omega_{t,n} \mid b(s_{t,n}, \nu_n),\ \psi_{t,n}).$$

These auxiliary variables are conditionally independent and hence can be updated in parallel. Moreover, efficient algorithms are available to generate Pólya-gamma random variates (Windle et al., 2014), and we have ported these to Python.*

5.2.3 Observation Parameter Updates

The observation parameter updates depend on the particular distribution. For Gaussian observations, ν_n is the observation variance, and it is conjugate with an inverse gamma prior. Bernoulli observations have no parameters. In the binomial model, ν_n corresponds to the maximum number of possible spikes — this may be treated as a hyperparameter. For negative binomial spike counts, the shape parameter ν_n can be sampled as in (Zhou et al., 2012).

5.2.4 Sampling Network Variables and Parameters

As before, the latent variables and parameters of the network are relatively easy to resample for a given network. The stochastic block model priors are conjugate with the Gaussian distributed weights. The locations of the latent distance model are not conjugate, but we can update them with hybrid Monte Carlo (Neal, 2010).

5.2.5 Model Selection

We have constructed a probabilistic model that supports a variety of network models, including the four adjacency models and the three weight models described above. How can we compare these models in a principled manner? We argue that the typical approach of measuring predictive log likelihood on held-out time bins is insufficient because it relies only on having accurately estimated the network, $\boldsymbol{A}$ and $\boldsymbol{W}$. It does not matter how likely that network is under the latent variable model, predictive likelihood on held-out time bins only measures the quality of the network at making predictions. Instead, we advocate for an alternative measure based on predicting the activity of held-out

*https://github.com/slinderman/pypolyagamma

neurons. To perform well on this task, we must first learn an accurate model for the structure underlying the network so that we can sample latent variables for the new neuron, which in turn allow us to sample a weighted set of functional connections for that neuron and finally compute the predictive log likelihood.

The objective we measure is the probability of a new spike train $\boldsymbol{s}_{n^*} = [s_{1,n^*}, \ldots, s_{T,n^*}]$, given the observed spike train. To compute this, we must integrate over the latent variables and parameters underlying the observed spike train, as well as those underlying the new spike train. Let $\boldsymbol{\Theta} = \{\{\boldsymbol{w}_n, \boldsymbol{a}_n, \nu_n, \boldsymbol{z}_n\}_{n=1}^N, \boldsymbol{\vartheta}\}$, and let $\boldsymbol{\theta}_{n^*} = \{\nu_{n^*}, \boldsymbol{w}_{n^*}, \boldsymbol{a}_{n^*}, \boldsymbol{z}_{n^*}\}$. This objective can be written,

$$\begin{aligned} p(\boldsymbol{s}_{n^*} \mid \boldsymbol{S}) &\approx \int p(\boldsymbol{s}_{n^*} \mid \boldsymbol{\theta}_{n^*}, \boldsymbol{S})\, p(\boldsymbol{\theta}_{n^*} \mid \boldsymbol{\Theta})\, p(\boldsymbol{\Theta} \mid \boldsymbol{S})\, \mathrm{d}\boldsymbol{\theta}_{n^*}\, \mathrm{d}\boldsymbol{Z} \\ &\approx \frac{1}{L} \sum_{\ell=1}^{L} p(\boldsymbol{s}_{n^*} \mid \boldsymbol{\theta}_{n^*}^{(\ell)}, \boldsymbol{S}), \end{aligned}$$

where

$$\boldsymbol{\theta}_{n^*}^{(\ell)} \sim p(\boldsymbol{\theta}_{n^*} \mid \boldsymbol{\Theta}^{(\ell)}), \qquad \text{and} \qquad \boldsymbol{\Theta}^{(\ell)} \sim p(\boldsymbol{\Theta} \mid \boldsymbol{S}).$$

The samples $\{\boldsymbol{\Theta}^{(\ell)}\}_{\ell=1}^L$ are the posterior samples generated by the MCMC algorithm presented above. For each sample, we generate a new set of latent variables and connections for neuron n^*, given the parameters included in $\boldsymbol{\Theta}^{(\ell)}$. These, combined with the spike train, enable us to compute the likelihood of $\boldsymbol{s}_{n^*}$.

This approach constitutes a minor approximation: the new spike train and the original spike train are not conditionally independent. It is possible that there are significant connections from n^* to neurons in the training population, and if we had known those connections, we would have inferred different latent variables and parameters for the training population. We assume that these effects are small, i.e. we would find similar class assignments even without observing n^*. This is reasonable if we are only considering a single neuron n^* and the training population is relatively large. In fact, this assumption is fundamental to the generalized linear model. Without it, the inferred weights and predictions would be highly sensitive to the addition of a single neuron. In practice this is rarely the case.

5.3 Results

We demonstrate the efficacy of this approach by applying our framework to two neural populations for which we have longstanding experimental evidence in favor of a particular latent variable representation. First, we consider a population of simultaneously recorded retinal ganglion cells (RGCs), which can be characterized by their type (either *on* or *off* in this dataset) and by the location of their receptive field centers. We then consider a population of hippocampal place cells, which encode positions in a two dimensional environment. In both cases, our approach recovers these latent representations given only the neural spike trains, without any knowledge of the stimulus or the true location. Finally, we assess the advantages of our Bayesian approach and the scalability of our algorithm with synthetic data.

5.3.1 Retinal Ganglion Cells

We applied our model to simultaneously recorded multi-neuronal spike trains from a population of 27 primate retinal ganglion cells (RGCs). This dataset was previously analyzed with a standard generalized linear model by Pillow et al. (2008). In this dataset, the population is presented with a Gaussian white noise video with pixel sizes tuned to approximately the width of a receptive field. We trained our models on one minute of spiking activity, binned at one millisecond time resolution. There were approximately 50,000 spikes in this recording. We also held out one minute of data for model evaluation.

Retinal ganglion cells respond to light (or the absence thereof) shown upon their receptive field. The cells are roughly evenly distributed across the two-dimensional retinal plane. Thus, it is natural to characterize these cells by the location of their receptive field center. Moreover, this population is comprised of two types of cells, *on* and *off* cells, characterized by their response to visual stimuli. *On* cells increase their firing when light is shone upon their receptive field; *off* cells decrease their firing rate in response to light in their receptive field. Cell types are identified by similarity in their response properties as well as morphological and physiological properties (Sanes and Masland, 2015). With knowledge of the stimulus, these cells can be clearly separated into their respective types by clustering the spike-triggered average stimulus. This characterization in terms of latent locations and cell types has been validated through decades of experiments (Kuffler, 1953), and has been made possible by the relative ease with which relevant stimuli can be identified and controlled. To what extent can these latent representations be discovered from the spike trains alone?

We fit our model to the measured spike trains for each of the twelve probabilistic network priors

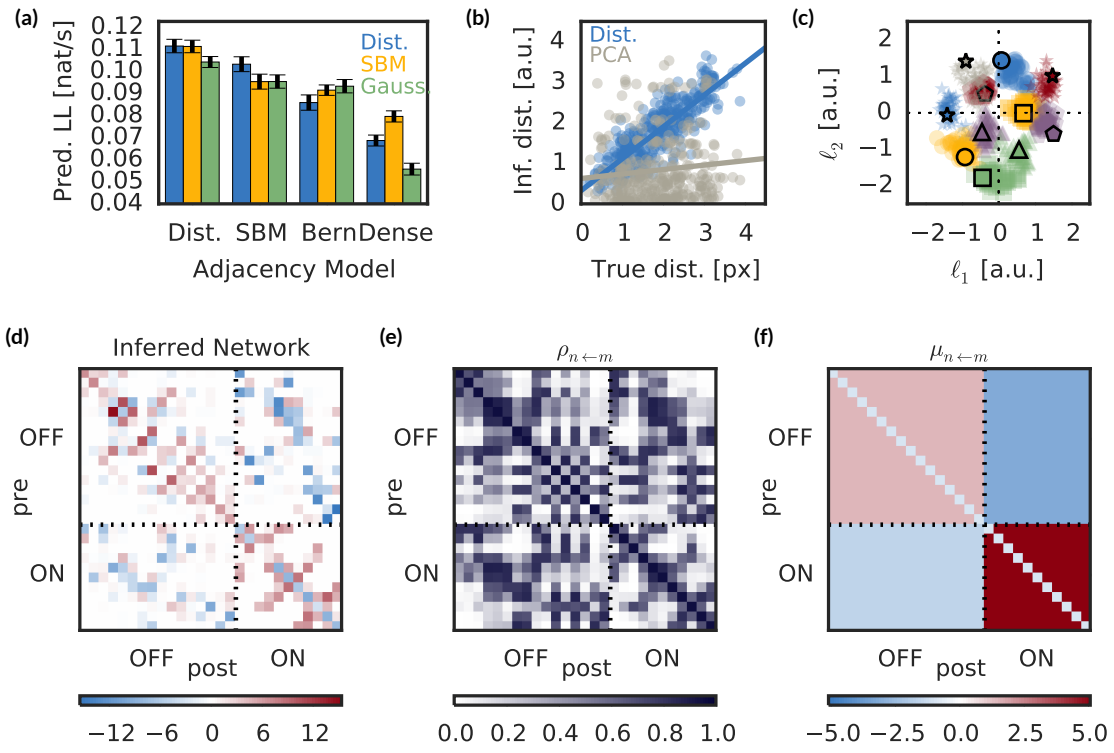

Figure 5.1: Retinal ganglion cell types and locations can be inferred from spike trains alone. **(a)** The combined distance model and block model yields the highest predictive log likelihood (units: nats/spike), compared to other combinations of adjacency models (groups of bars) and weight models (*blue*: latent distance weight model, *yellow*: stochastic block model for weights, *green*: Gaussian weights). **(b)** The inferred distances under the latent distance model are highly correlated with the true distances, as measured by stimulus sensitivity, whereas inferred embeddings under PCA are not. **(c)** Moreover, the inferred locations (semi-transparent markers) recover the true receptive field centers (solid markers with black outline). Shown here only for ON cells. **(d)** Inferred network, $\boldsymbol{A} \odot \boldsymbol{W}$, under a latent distance model of connection probability and a stochastic block model for connection weight, averaged over 500 posterior samples. **(e)** Expected probability of connection under the latent distance model. **(f)** Expected connection strength under the stochastic block model. The inferred cell classes perfectly match the true ON and OFF types, and reflect within-class excitation and between-class inhibition.

shown in Fig. 3.2 — three weight models and four adjacency models. Predictive likelihood comparisons on the held-out data reveal that the adjacency matrix, i.e. the pattern of connectivity, is well-characterized by a two-dimensional latent distance model (Fig. 5.1c), as expected given the localized receptive fields of RGCs. Moreover, the pairwise distances between the latent locations are highly correlated with the true distances between receptive field centers (Fig. 5.1b), even though the stimulus was never used during training. By contrast, the inferred locations given by the top two

principal components of the spike trains are highly uncorrelated, indicating that PCA does not recover a meaningful spatial embedding. Finally, the inferred locations can be rotated and scaled such that they match the true locations almost perfectly (Fig. 5.1c). Rotation does not change the pairwise distances and scaling simply changes the units. This matching is highly unlikely for randomly distributed locations. In Fig. 5.1c, the true locations are shown as solid markers with black outlines, and the inferred locations sampled from the posterior distribution are shown as semi-transparent markers of the same color and shape.

For the weight matrix, a latent distance model (Dist) and a stochastic block model (SBM) both yield similar predictive likelihoods. Looking into the inferred types under the SBM, we find that the neurons are grouped according to their true *on* and *off* cells, again without any knowledge of the stimulus. These types determine the expected interaction weight for each pair of neurons.

Figure 5.1d shows the inferred functional network under the latent distance adjacency model and stochastic block model for weights. The neurons are sorted first by their type (*off* then *on*) and then by their x-location. This yields the three bands in the matrix of connection probabilities 5.1e. Nearby cells have much higher probability of connection. The mean connection strength (Fig. 5.1f) shows the characteristic pattern of positive weights between cells of the same type and negative weights between cells of opposite types. The diagonal of this matrix shows the weights of self-connections, which are typically negative due to refractory effects.

Together, these inferred types and locations provide compelling evidence for a highly structured pattern of functional connectivity. Given the extensive work on characterizing retinal ganglion cell responses, we have considerable evidence that the representation we learn from spike trains alone is indeed the optimal way to characterize this population of cells. This lends us confidence that we may trust the representations learned from spike trains recorded from deeper brain areas, where traditional stimulus-response correlations are less valuable.

5.3.2 Hippocampal Place Cells

We also applied our framework to simultaneously recorded multi-neuronal spike trains from the hippocampal recordings studied in Chapter 3. Here, however, we preprocess the data by binning the spikes into 250ms time bins. Furthermore, we only consider the spike counts from time bins in which the rat was moving at a velocity of at least 10cm/s, since these hippocampal place cells fire primarily during locomotion. The resulting dataset is almost ten minutes long. Finally, we only consider the 25 neurons with the most precise place fields.

As in the retina, we have strong intuitions about the latent structure underlying hippocampal

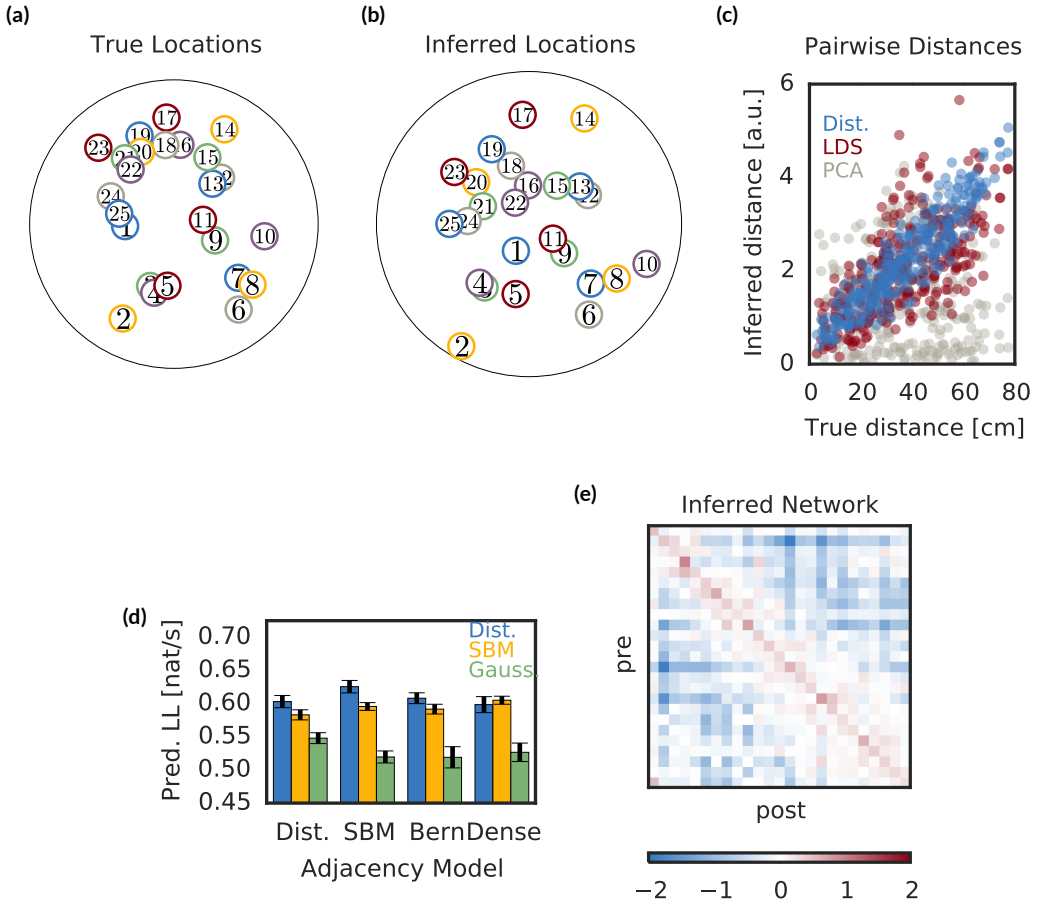

Figure 5.2: Hippocampal place fields are inferred from population spike trains.

place cell activity, namely, we expect cells representing nearby locations to be correlated and cells with disjoint place fields to be anti-correlated. Can we extract the spatial layout of place fields from spike trains alone?

We apply our framework, fitting all twelve network models and evaluating them on the basis of predictive likelihood. We use a negative binomial observation model to allow multiple spikes per time bin. We find that the distance dependent weight models yield the highest predictive likelihood (Fig. 5.2d). The stochastic block model yields the highest predictive likelihood, but upon further investigation we find that almost all neurons are connected in this model. There is one outlier, neuron #1, which is only sparsely connected. The SBM assigns 24 neurons are assigned to one cluster, and assigns neuron #1 to its own group. Thus, the stochastic block model is quite similar to an independent Bernoulli model, as is evident from the similar predictive likelihood of these two models.

Looking into the inferred locations under the latent distance model for the weight matrix, we find that the inferred locations (Fig 5.2b) are (up to rotation and scale) highly similar to the true place field centers (Fig 5.2a) measured using the rat's true location. This is quantified by plotting the pairwise distances between inferred locations against the pairwise distances between place field centers. The latent distance model's pairwise distances are highly correlated with the ground truth, whereas distances between PCA embeddings are very uncorrelated (Fig. 5.2c).

For further comparison, we fit a Poisson linear dynamical system (PLDS) (Macke et al., 2011) with a two dimensional latent state space, and used the rows of the $N \times 2$ emission matrix as the locations of the neurons. Chapter 8 will discuss this class of models in more detail. The PLDS yields pairwise distances that are quite correlated with the true distances, but we find that the variance of this fit grows with the true distance. This reflects the fact that the PLDS does not explicitly parameterize interaction as a function of distance, but rather induces correlation due to similarity in the emission matrix as well as shared dynamics.

Finally, we investigated the expected network under the posterior and found that upon sorting the neurons by their inferred locations, an intuitive banded structure emerges (Fig. 5.2e). The positive diagonal indicates strong autocorrelation between a neurons spiking in consecutive 250ms bins. Over these time scales, self-refractory effects are not evident. The primary connections are inhibitory in nature, such that active cells suppress the activity of cells with distal place fields.

These inferred representations again confirm our intuitive beliefs about the structure of hippocampal place cell responses. The inferred latent variables recover meaningful structure in the neural activity, without any access to the location. Instead, these representations arise from the neural activity alone.

5.3.3 Synthetic Data

To assess the robustness and scalability of our framework, we apply our methods to simulated data with known ground truth. First, we show that our Bayesian approach can recover structure in larger populations of neurons than those studied above. Moreover, we show that our approach outperforms alternative methods that separate the problems of finding the network and discovering the underlying structure. Then, we show that for sparse networks, our approach only scales quadratically with the number of neurons — a significant improvement over naïve methods that incur an additional $O(N^3)$ cost per potential connection.

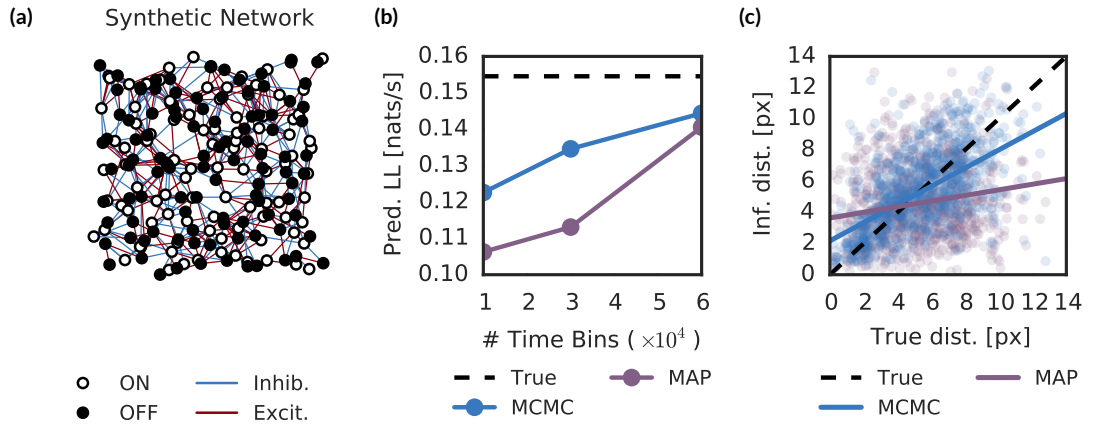

Figure 5.3: Synthetic results based on the retinal ganglion cell population studied above. **(a)** Locations of the 200 neurons (100 *on* and 100 *off* cells) along with a subset of the connections. For clarity, 15% of the connections are randomly chosen. Though the network is not symmetric, arrowheads are dropped for clarity. **(b)** Predictive likelihood on 10,000 held-out time bins as a function of the number of time bins in the training dataset. "MCMC" denotes our Bayesian approach with a latent distance adjacency model and a SBM weight model. "MAP" denotes the L_1-regularized standard GLM. **(c)** Inferred locations found by our model (MCMC) and by fitting a latent distance model to the thresholded standard GLM network (MAP).

Advantages of being Bayesian

For these analyses, we simulate a population of 200 neurons with structure that mimics that of the retinal ganglion cell population analyzed in Section 5.3.1. Figure 5.3a illustrates the underlying network. As before, we have *on* cells and *off* cells, each centered at a location in the 2D retinal plane. Nearby cells are more likely to be functionally connected, and cells of the same type will excite each other whereas cells of different types will inhibit one another. Again, these functional connections reflect correlations and anti-correlations between cells that arise from common input. We simulate 60000 time bins and tune the network parameters such that if each time bin were one millisecond, the firing rates would range from 10Hz to 70Hz with a mean of about 30Hz. Rather than simulating an external white noise stimulus, here we simulate spontaneous activity of the network.

Since we know the generative model that gave rise to the data, we can quantify the improvement from using our fully Bayesian approach rather than a standard GLM, and compare these improvements to the upper bound provided by the true model. Figure 5.3b shows the predictive log likelihood for 10000 bins of held-out data as a function of the number of time bins of training data. For shorter training datasets, our Bayesian approach yields considerably higher predictive likelihood than the standard, L_1-regularized GLM. As the length of the training data increases, both our

method and the standard GLM improve, and ultimately approach the likelihood of the true model that generated the data. With more data, the network inference is driven less by the prior and more by the observations. Thus, in the limit of infinite data,the GLM will converge to the same network, and hence the same predictive likelihood, as our Bayesian method. However, with hundreds of neurons and tens of thousands of time bins, these results suggest that the Bayesian approach can yield substantial benefits.

Rather than simultaneously fitting the network and the underlying latent variables, one may instead fit the network with an L_1-regularized GLM and *then* fit a probabilistic network model to the GLM connection weights. This is the approach taken by Stevenson et al. (2009). The advantage is that the network is only fit once, and since this is usually the most expensive aspect of inference, it can save considerable time. However, when the data is limited, both the network and the latent variables are uncertain. Our Bayesian approach finds joint assignments with high posterior probability, whereas the standard GLM finds a single network that does not account for prior beliefs about structure underlying the network. In this example, subsequently fitting a latent distance model to the adjacency matrix of a thresholded GLM network finds an embedding that differs considerably from the embedding found by our Bayesian approach. This is illustrated by the decreased correlation between true and inferred pairwise distances under the standard GLM compared to that of our joint, Bayesian approach, as shown in Figure 5.3c.

Figure 5.4 further illustrates this point. Here, the true network is shown as a weighted adjacency matrix (Fig. 5.4a), along with its binary adjacency matrix (Fig. 5.4d). The inferred network and connection probabilities from our fully Bayesian method with a latent distance model prior are shown in Fig. 5.4b and Fig. 5.4e, respectively. Our MCMC algorithm does a good job recovering the underlying locations, as shown by the diagonally banded connection probabilities reflecting the distance dependence. By contrast, the MAP network found via an L_1-regularized standard GLM is considerably noisier (Fig. 5.4c), since it knows nothing of the underlying distance dependent connectivity. A latent distance model fit to a thresholded copy of this network fails to capture the distance dependence (Fig. 5.4f), since the spurious connections force otherwise well-separated neurons to lie close in the embedding. By jointly fitting the network and the latent locations, we weed out these spurious connections.

Scalability Analysis

Finally, we address the scalability of our Bayesian inference algorithms. With large-scale recordings becoming the norm, efficiency is paramount. There are three major parameters that govern the com-

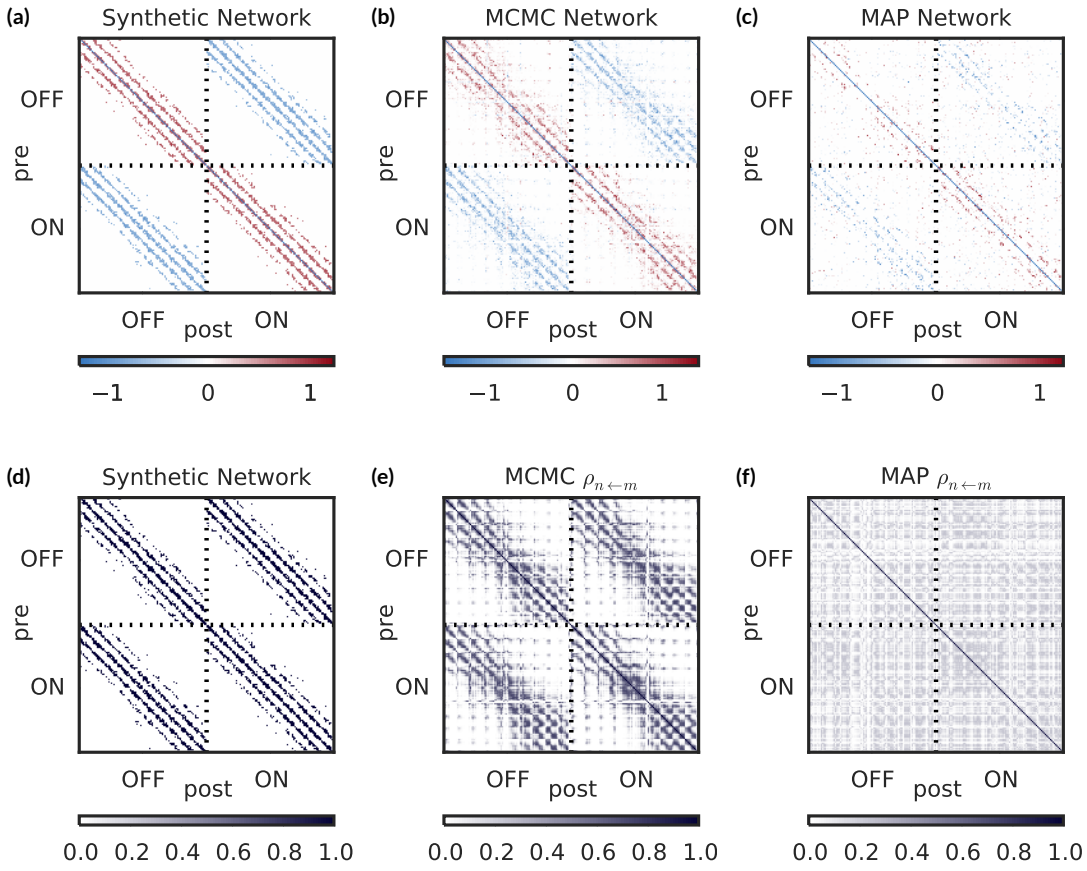

Figure 5.4: **(a)** Weighted adjacency matrix for the synthetic RGC network shown in Fig. 5.3a. **(b)** Network inferred by our fully Bayesian approach. **(c)** Thresholded network found by an L_1-regularized GLM. **(d)** Adjacency matrix corresponding to the true network in Fig. 5.4a. **(e)** Inferred connection probability found by our model with a latent distance prior. The diagonal bands reflect the increased probability for nearby neurons.. **(f)** Inferred connection probability found by fitting a latent distance model to the network in Fig. 5.4c. Noise in this network leads to poor location estimates and nearly uniform connection probabilities.

plexity of inference: the number of time bins, T; the number of neurons, N; and the level of sparsity, ρ. The natural unit of measure is the wall-clock time required to perform one iteration of our MCMC algorithm. The following experiments were run on a quad-core Intel i5 with 6GB of RAM. We break down the wall-clock time into time spent updating the auxiliary variables of the observation model and time spent updating the network. Resampling the latent variables incurs a lesser cost.

The wall clock time scales linearly with T, as shown in Figure 5.5a. Recall that the observation

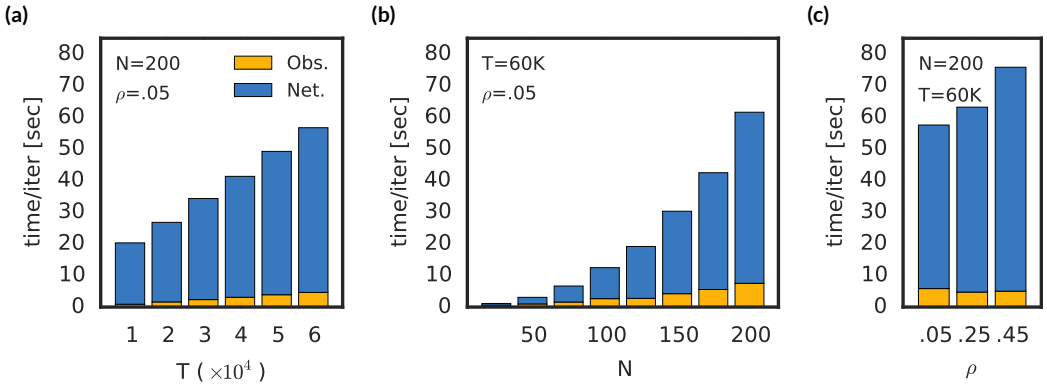

Figure 5.5: Scalability of our inference algorithm as a function of: **(a)** the number of time bins, T; **(b)** the number of neurons, N; and **(c)** the average sparsity of the network, ρ. The complexity grows linear with T and quadratically with N (for fixed ρ). The cost is broken down into the cost of resampling the Pólya-gamma auxiliary variables ("Obs.") and the cost of resampling the sparse network ("Net."). Color scheme shared across all panels.

model contains NT auxiliary variables, each of which must be resampled. Additionally, each neuron must compute its sufficient statistics, which involves a rectangular matrix multiplication costing $O(TN^2)$ time.

Since there are $O(N^2)$ possible connections in a network of N neurons, there is no way to avoid at least a quadratic cost in N (unless some connections can be ruled out *a priori*). Figure 5.5b shows that this quadratic penalty is indeed evident. The cost of updating the auxiliary variables is only linear in N, but the cost of updating the network is quadratic. In Section 5.4, we discuss potential strategies for ruling out connections and limiting this cost.

The total cost could actually be worse than quadratic because the cost of updating each connection could depend on N. Fortunately, the complexity of our collapsed Gibbs sampling algorithm only depends on the number of incident connections received by each neuron, p. Specifically, we must invert a $p \times p$ matrix, which incurs an $O(p^3)$ cost. Figure 5.5c illustrates this for a Bernoulli network with $N = 200$ neurons. The expected number of incoming connections is related to the probability of connection by $\mathbb{E}[p] = \rho N$. If we increased the number of neurons but kept the average in-degree constant, the total cost would scale as $\mathcal{O}(TN + N^2 p^3)$.

5.4 Discussion

We have shown that the functional networks underlying spike trains may provide insight into the low-dimensional structure of neural populations. Looking forward, there are two main concerns that we would like to address. First, ideally we would like algorithms that scale better than quadrat-

ically with the number of neurons. Second, we would like to automate the process of relating the latent structure to simultaneously measured environmental cues, stimuli, and behavioral correlates.

5.4.1 Further scalability improvements

For large populations, the bulk of the running time is spent sampling the network variables, $\boldsymbol{A}$ and $\boldsymbol{W}$. As the number of neurons grows, this complexity scales as $O(N^2)$. To what extent can we minimize this complexity? As we have shown, naïvely applying clustering algorithms and PCA directly to spike trains does not necessarily yield meaningful features. These results can be somewhat improved by smoothing the spike trains (e.g. by taking a moving average), but they are still fundamentally limited when the low-dimensional structure lies in the pattern of functional connectivity.

Moreover, our results in Section 5.3.3 show that separating the problem of inferring the network and discovering its latent structure may yield to inferior results. This is particularly common for large neural populations, where there is substantial posterior uncertainty. In practice, we initialize our method with a standard GLM and find that a small number of iterations of our MCMC algorithm can yield large improvements in the predictive likelihood of our model. These improvements come from pruning connections that are weakly supported by the data, but which are very unlikely under the prior. This improvement in predictive likelihood is consistent with existing literature on Bayesian spike-and-slab modeling (Mohamed et al., 2012).

An alternative approach is to consider models that explicitly limit the dimensionality of the network. For example, if we know the physical location of neurons (or their approximate location from electrode number), we may rule out long-range connections. Alternatively, we may constrain the weighted adjacency matrix to be low rank, that is $\boldsymbol{W} \odot \boldsymbol{A} \equiv \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are rank $K < N$. The effect is similar to that of a linear dynamical system model (Paninski et al., 2010; Macke et al., 2011). Unfortunately, it is less clear how to incorporate interpretable structure into this type of model, although see Buesing et al. (2014) for some initial steps in this direction.

Finally, we have developed a Markov chain Monte Carlo inference algorithm, and while we have shown reasonable performance on recordings of hundreds of neurons over tens of thousands of time bins, as we scale to larger recordings alternative algorithms may be preferable. When the complexity is dominated by the number of time bins, stochastic variational inference (SVI) (Hoffman et al., 2013) provides an attractive alternative. Since the time bins are conditionally independent given the network, we can get unbiased estimates of sufficient statistics from *mini-batches* of time bins and use them to inform stochastic mean field updates. While we have not derived this mean field algorithm here, it is relatively straightforward to derive these algorithms for Pólya-gamma aug-

mented models (Pillow and Scott, 2012; Zhou et al., 2012). Still, the limiting factor will often be the number of neurons, in which case SVI is not immediately applicable. Developing approximate inference algorithms that only consider subsets of neurons is an active area of research (Soudry et al., 2015).

5.4.2 Relating circuit structure to stimuli and behavior

So far we have reserved the externally measured covariates for validation. We used the white noise stimulus in the retinal ganglion cell experiment to determine the neurons' receptive fields, and we used the measured location in the hippocampal experiment to determine the place cell locations. We did not, however, attempt to jointly model the neural activity *and* the external covariates, as is often done in GLM analyses. The simplest way to incorporate these external covariates, $\boldsymbol{Y} = \{\boldsymbol{y}_t\}$, into the model is via a linear term in the activation. That is, let,

$$\psi_{t,n} \triangleq \psi_n^{(0)} + \boldsymbol{u}_n^\mathsf{T} \boldsymbol{y}_t + (\boldsymbol{a}_n \odot \boldsymbol{w}_n)^\mathsf{T} \widehat{\boldsymbol{s}}_t,$$

where $\boldsymbol{u}_n$ is the "stimulus filter." Just as we modeled $\boldsymbol{a}_n$ and $\boldsymbol{w}_n$ with hierarchical network models, we may model $\boldsymbol{u}_n$ in terms of a set of neuron-specific latent variables.

As we explore deeper brain circuits that are further removed from sensory inputs or motor outputs, the relationship between neural activity and external covariates becomes less clear. In these cases, the unsupervised methods for extracting structured representations can provide useful hints as to how neural activity may be understood. That the representations found in these well-studied circuits of the retina and the hippocampus match our expectations provides reason to believe that they can be fruitfully applied to more enigmatic circuits as well.

5.5 Conclusion

This chapter extended the linear autoregressive models of previous chapters to include both excitatory and inhibitory interactions by developing a nonlinear firing rate model. We derived an efficient, fully-conjugate Gibbs sampling algorithm that integrate out the weights in order to sample sparsity patterns. The key was the Pólya-gamma augmentation, which makes discrete observations appear as Gaussian likelihoods. This led to dramatic improvements in performance and allowed us to discover interesting structure in a variety of real and synthetic datasets.

Nevertheless, these methods are still limited in that they are parameterized in terms of stationary network. There is no aspect of the model that captures time-varying properties of the neural circuit

under study. In the next chapter, we consider extensions of this model that capture dynamics in the network itself. This provides a step toward dynamical systems models that attempt to explain data in terms of a latent state that evolves over time.

6

Dynamic Network Models

Synaptic plasticity is believed to be the fundamental building block of learning and memory in the brain (Dayan and Abbott, 2001). Its study is of crucial importance to understanding the activity and function of neural circuits. With innovations in neural recording technology providing access to the simultaneous activity of increasingly large populations of neurons, statistical models are promising tools for formulating and testing hypotheses about the dynamics of synaptic connectivity. Advances in optical techniques (Packer et al., 2012; Hochbaum et al., 2014), for example, have made it possible to simultaneously record from and stimulate large populations of synaptically connected neurons. Armed with statistical tools capable of inferring time-varying synaptic connectivity, neuroscientists could test competing models of synaptic plasticity, discover new learning rules at the mono-synaptic and network level, investigate the effects of disease on synaptic plasticity, and potentially design stimuli to modify neural networks.

Despite the popularity of autoregressive models for spike data, like the GLM (Paninski, 2004; Truccolo et al., 2005; Pillow et al., 2008), relatively little work has attempted to model the time-varying nature of neural interactions. Here we model interaction weights as a dynamical system governed by parametric synaptic plasticity rules. Building on the work of preceding chapters, we show how synaptic plasticity rules can be modeled as dynamics rules that govern how weights evolve in an activity-dependent manner. In doing so, we imbue the weights with a biophysical interpretation that we explicitly avoided in previous chapters. We discuss when this interpretive leap is warranted.

To perform inference in this model, we use particle Markov chain Monte Carlo (pMCMC) (Andrieu et al., 2010), a recently developed inference technique for time series with nonlinear dynamics. We use this new modeling framework to examine the problem of using recorded data to distinguish between proposed variants of spike-timing-dependent plasticity (STDP) learning rules. On synthetic data generated from the biophysical simulator NEURON, we show that we can recover the weight trajectories, the pattern of connectivity, and the underlying learning rules.

6.1 A Biophysical Interpretation of the GLM

The nonlinear autoregressive models of the previous chapter treat the spike count, $s_{t,n}$, as a random variable whose distribution depends on a nonnegative firing rate, $\lambda_{t,n}$. The firing rate is modeled as a nonlinear function of an activation, $\psi_{t,n}$, which is taken to be a linear function of the spike history. This linear-nonlinear cascade is often called a generalized linear model (GLM) (Paninski, 2004; Truccolo et al., 2005).

From a biophysical perspective, the activation can be thought of as analogous to the cell's membrane potential. The nonlinearity that links the activation to the firing rate approximates the spiking threshold of the neuron. When the membrane potential exceeds the spiking threshold potential of the cell, $\lambda_{t,n}$ rises to reflect the rate of the cell's spiking, and when the membrane potential decreases below the spiking threshold, $\lambda_{t,n}$ decays to zero.

As before, we model the activation, or membrane potential, as a linear function of the spike history,

$$\psi_{t,n} = \psi_n^{(0)} + \sum_{n=1}^{N} \sum_{d=1}^{D} h_{n' \to n}[d] \cdot s_{t-d,n'}. \tag{6.1}$$

where $\psi_n^{(0)}$ is now the resting potential and $h_{n' \to n}[d]$ is a post-synaptic potential that preceding spikes on neuron n' induce on the membrane potential of neuron n at lag d.

From this semi-biophysical perspective it is clear that one shortcoming of the models developed thus far is that they do not account for time-varying connectivity, despite decades of research showing that changes in synaptic weight occur over a variety of time scales and are the basis of many fundamental cognitive processes. This absence is due, in part, to the fact that this direct biophysical interpretation is not warranted in most traditional experimental regimes, e.g., in multi-electrode array (MEA) recordings where electrodes are relatively far apart. However, as high resolution optical recordings grow in popularity, this assumption must be revisited; this is a central motivation for the

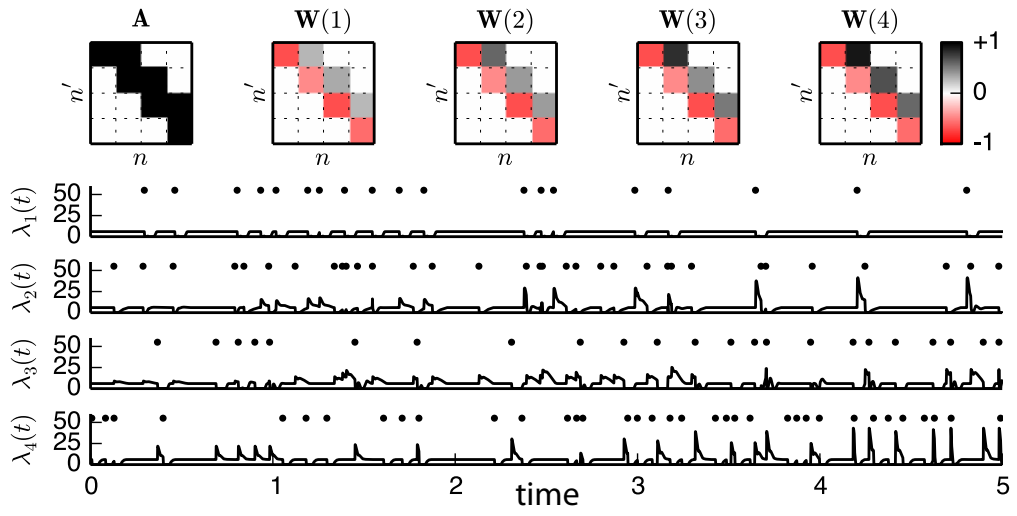

Figure 6.1: A simple network of four sparsely connected neurons whose synaptic weights are changing over time. Here, the neurons have inhibitory self connections to mimic refractory effects, and are connected via a chain of excitatory synapses, as indicated by the nonzero entries $a_{1\to 2}$, $a_{2\to 3}$, and $a_{3\to 4}$. The corresponding weights of these synapses are strengthening over time (darker entries in $\boldsymbol{W}$), leading to larger impulse responses in the firing rates and a greater number of induced post-synaptic spikes (black dots), as shown below.

model developed in this chapter.

There have been a few efforts to incorporate dynamics into the GLM. Stevenson and Koerding (2011) extended the GLM to take inter-spike intervals as a covariates and formulated a generalized bilinear model for weights. Eldawlatly et al. (2010) modeled the time-varying parameters of a GLM using a dynamic Bayesian network (DBN). However, neither of these approaches accommodate the breadth of synaptic plasticity rules present in the literature. For example, parametric STDP models with hard bounds on the synaptic weight are not congruent with the convex optimization techniques used by (Stevenson and Koerding, 2011), nor are they naturally expressed in a DBN. Here we model time-varying synaptic weights as a potentially nonlinear dynamical system and perform inference using particle MCMC.

Nonstationary, or time-varying, models of synaptic weights have also been studied outside the context of GLMs. For example, Petreska et al. (2011) applied hidden switching linear dynamical systems models to neural recordings. This approach has many merits, especially in traditional MEA recordings where synaptic connections are less likely and nonlinear dynamics are not necessarily warranted. Outside the realm of computational neuroscience and spike train analysis, there exist a number of dynamic statistical models, such as the dynamic generalized linear models of West et al.

(1985). However, the types of models we are interested in for studying synaptic plasticity are characterized by domain-specific transition models and sparsity structure, and until recently, the tools for effectively performing inference in these models have been limited.

6.2 A Sparse Time-Varying Generalized Linear Model

In order to capture the time-varying nature of synaptic weights, we extend the standard GLM by first factoring the impulse responses in the firing rate of (6.1) into a product of three terms:

$$h_{n' \to n}[d, t] \triangleq a_{n' \to n} \cdot w_{n' \to n}[t] \cdot \hbar_{n' \to n}[d]. \tag{6.2}$$

Here, $a_{n' \to n} \in \{0, 1\}$ is a binary random variable indicating the presence of a direct synapse from neuron n' to neuron n, $w_{n' \to n}[t] \in \mathbb{R}$ is a non stationary synaptic "weight" trajectory associated with the synapse, and $\hbar_{n' \to n}[d]$ is a nonnegative, normalized impulse response, i.e. $\sum_{d=1}^{D} \hbar_{n' \to n}[d] \cdot \Delta t = 1$. Requiring $\hbar_{n' \to n}[d]$ to be normalized gives meaning to the synaptic weights: otherwise w would only be defined up to a scaling factor. For simplicity, we assume $\hbar[d]$ does not change over time, that is, only the amplitude and not the duration of the PSPs is time-varying. This restriction could be adapted in future work.

As in previous chapters, we model the normalized impulse responses as a linear combination of basis functions. In order to enforce the normalization of $\hbar[d]$, however, we use a *convex* combination of normalized, nonnegative basis functions. That is,

$$\hbar_{n' \to n}[d] \equiv \sum_{b=1}^{B} \theta^{(b)}_{n' \to n} \, \phi_b[d],$$

where $\sum_{d=1}^{D} \phi_b[d] \cdot \Delta t = 1$ and $\sum_{b=1}^{B} \theta^{(b)}_{n' \to n} = 1$.

The binary random variables $a_{n' \to n}$, which can be collected into an $N \times N$ binary matrix $\boldsymbol{A}$, model the connectivity of the synaptic network. Similarly, the collection of weight trajectories $\{\{w_{n' \to n}[t]\}\}_{n',n}$, which we will collectively refer to as $\boldsymbol{W}[t]$, model the time-varying synaptic weights. This factorization is often called a *spike-and-slab* prior (Mitchell and Beauchamp, 1988), and it allows us to separate our prior beliefs about the structure of the synaptic network from those about the evolution of synaptic weights. For example, in the most general case we might incorporate the probabilistic network models of previous chapters as prior distributions for $\boldsymbol{A}$, but here we limit ourselves to the simplest network model, the independent Bernoulli, or Erdős-Rényi model. Under

this model, each $a_{n' \to n}$ is an independent identically distributed Bernoulli random variable with sparsity parameter ρ.

Figure 6.1 illustrates how the adjacency matrix and the time-varying weights are integrated into the GLM. Here, a four-neuron network is connected via a chain of excitatory synapses, and the synapses strengthen over time due to an STDP rule. This is evidenced by the increasing amplitude of the impulse responses in the firing rates. With larger synaptic weights comes an increased probability of post-synaptic spikes, shown as black dots in the figure. In order to model the dynamics of the time-varying synaptic weights, we turn to a rich literature on synaptic plasticity and learning rules.

6.2.1 Learning Rules for Time-Varying Synaptic Weights

Decades of research on synapses and learning rules have yielded a plethora of models for the evolution of synaptic weights (Caporale and Dan, 2008). In most cases, this evolution can be written as a dynamical system,

$$\boldsymbol{W}[t+1] = \boldsymbol{W}[t] + \ell\left(\boldsymbol{W}[t],\, \boldsymbol{S}_{\leq t}, \boldsymbol{\vartheta}\right) + \epsilon(\boldsymbol{W}[t], \boldsymbol{\vartheta})$$

where ℓ is a potentially nonlinear *learning rule* that determines how synaptic weights change as a function of previous spiking, $\boldsymbol{S}_{\leq t}$. This framework encompasses rate-based rules such as the Oja rule (Oja, 1982) and timing-based rules such as STDP and its variants. The additive noise, $\epsilon(\boldsymbol{W}[t], \boldsymbol{\vartheta})$, need not be Gaussian, and many models require truncated noise distributions.

Following biological intuition, many common learning rules factor into a product of simpler functions. For example, STDP (defined below) updates each synapse independently such that the learning rule for $w_{n' \to n}$ only depends on the current weight, $w_{n' \to n}[t]$, and the pre- and post-synaptic spike history, $\boldsymbol{S}_{\leq t}$. Biologically speaking, this means that plasticity is local to the synapse. More sophisticated rules allow dependencies among the columns of $\boldsymbol{W}$. For example, the incoming weights to neuron n may depend upon one another through normalization, as in the Oja rule (Oja, 1982), which scales synapse strength according to the total strength of incoming synapses.

Extensive research in the last fifteen years has identified the relative spike timing between the pre- and post-synaptic neurons as a key component of synaptic plasticity, among other factors such as mean firing rate and dendritic depolarization (Feldman, 2012). STDP is therefore one of the most prominent learning rules in the literature today, with a number of proposed variants based on cell type and biological plausibility. In the experiments to follow, we will make use of two of these pro-

posed variants. First, consider the canonical STDP rule with a "double-exponential" function parameterized by $\boldsymbol{\vartheta} = \{\tau_-, \tau_+, A_-, A_+\}$ (Song et al., 2000), in which the effect of a given pair of pre-synaptic and post-synaptic spikes on a weight may be written:

$$\ell\left(w_{n'\to n}[t], \boldsymbol{S}_{\leq t}; \boldsymbol{\vartheta}\right) = \ell_+(\boldsymbol{S}_{\leq t}, A_+, \tau_+) - \ell_-(\boldsymbol{S}_{\leq t}, A_-, \tau_-), \tag{6.3}$$

where,

$$\ell_+(\boldsymbol{S}_{\leq t}, A_+, \tau_+) = s_{t,n} \sum_{t'=1}^{t} s_{t',n'} \cdot A_+ \cdot e^{(t-t')/\tau_+},$$
$$\ell_-(\boldsymbol{S}_{\leq t}, A_-, \tau_-) = s_{t,n'} \sum_{t'=1}^{t} s_{t',n} \cdot A_- \cdot e^{(t-t')/\tau_-}.$$

This rule states that weight changes only occur at the time of pre- or post-synaptic spikes, and that the magnitude of the change is a nonlinear function of inter-spike intervals.

A slightly more complicated model known as the multiplicative STDP rule extends this by bounding the weights above and below by W_{max} and W_{min}, respectively (Morrison et al., 2008). Then, the magnitude of the weight update is scaled by the distance from the threshold:

$$\begin{aligned}\ell\left(w_{n'\to n}[t], \boldsymbol{S}_{\leq t}, \boldsymbol{\vartheta}\right) = \tilde{\ell}_+(\boldsymbol{S}_{\leq t}, A_+, \tau_+)\ (W_{\mathsf{max}} - w_{n'\to n}[t]),\\ - \tilde{\ell}_-(\boldsymbol{S}_{\leq t}, A_-, \tau_-)\ (w_{n'\to n}[t] - W_{\mathsf{min}}).\end{aligned} \tag{6.4}$$

Here, by setting $\tilde{\ell}_\pm = \min(\ell_\pm, 1)$, we enforce that the synaptic weights always fall within $[W_{\mathsf{min}}, W_{\mathsf{max}}]$. With this rule, it often makes sense to set W_{min} to zero.

Similarly, we can construct an additive, bounded model which is identical to the standard additive STDP model except that weights are thresholded at a minimum and maximum value. In this model, the weight never exceeds its set lower and upper bounds, but unlike the multiplicative STDP rule, the proposed weight update is independent of the current weight except at the boundaries. In the canonical STDP model it is sensible to use Gaussian noise, but in the bounded multiplicative model we use truncated Gaussian noise to respect the hard upper and lower bounds on the weights. Note that this noise is dependent upon the current weight, $w_{n'\to n}[t]$.

The nonlinear nature of this rule, which arises from the multiplicative interactions among the parameters, $\boldsymbol{\vartheta} = \{A_+, \tau_+, A_-, \tau_-, W_{\mathsf{max}}, W_{\mathsf{max}}\}$, combined with the potentially non-Gaussian noise models, pose substantial challenges for inference. However, the computational cost of these

detailed models is counterbalanced by dramatic expansions in the flexibility of the model and the incorporation of *a priori* knowledge of synaptic plasticity. These learning models can be interpreted as strong regularizers of models that would otherwise be highly under-determined, as there are N^2 weight trajectories and only N spike trains. In the next section we will leverage powerful new techniques for Bayesian inference in order to capitalize on these expressive models of synaptic plasticity.

6.3 Inference via Particle MCMC

The traditional approach to inference in the standard GLM is penalized maximum likelihood estimation. For a model with Poisson observations and a nonlinear link function, $g : \mathbb{R} \to \mathbb{R}_+$, the log likelihood is,

$$\log p(\boldsymbol{S} \mid \boldsymbol{\Lambda}) = \sum_{n=1}^{N} \sum_{t=1}^{T} -\lambda_{t,n} \, \Delta t + s_{t,n} \, \log \lambda_{t,n} \tag{6.5}$$

$$= \sum_{n=1}^{N} \sum_{t=1}^{T} -g(\psi_{t,n}) \, \Delta t + s_{t,n} \, \log g(\psi_{t,n}) \tag{6.6}$$

and the log likelihood of a population of non-interacting spike trains is simply the sum of each of the log likelihoods for each neuron. The likelihood depends upon the network through the definition of the activation given in Eq. 6.1 and Eq. 6.2.

Due to the potentially nonlinear and non-Gaussian nature of STDP, these existing techniques are not applicable here. Instead we use particle MCMC (Andrieu et al., 2010), a powerful technique for inference in time series. Particle MCMC samples the posterior distribution over weight trajectories, $\boldsymbol{W}[t]$, the adjacency matrix $\boldsymbol{A}$, and the model parameters $\boldsymbol{\theta}^{(n \to n')}$ and $\boldsymbol{\vartheta}$, given the observed spike trains, by combining particle filtering with MCMC. We represent the conditional distribution over weight trajectories with a set of discrete particles, $\{\boldsymbol{W}^{(p)}\}_{p=1}^{P}$. Each particle represents a sequence of weight matrices, $\boldsymbol{W}^{(p)} \in \mathbb{R}^{N \times N \times T}$, and has an associated nonnegative *particle weight* $v_T^{(p)}$. Note that the particle weights are *not* the same as the synaptic weights. Together, these define an atomic distribution over weight trajectories,

$$p(\boldsymbol{W}) \approx \frac{\sum_{p=1}^{P} v_T^{(p)} \, \delta_{\boldsymbol{W}^{(p)}}(\boldsymbol{W})}{\sum_{p=1}^{P} v_T^{(p)}}, \tag{6.7}$$

where $\delta_{w*}(w)$ is the Dirac delta function located at w^*.

6.3.1 Particle Filtering

Particle filtering (Andrieu et al., 2003) or *sequential Monte Carlo* is a method of inferring a distribution over weight trajectories by iteratively propagating forward in time and re-weighting according to how well the new samples explain the data. We build up the collection of weight trajectories iteratively, one bin at a time. We start by sampling the initial synaptic weights from the prior distribution,

$$\boldsymbol{W}^{(p)}[1] \sim p(\boldsymbol{W}[1] \mid \boldsymbol{\vartheta}),$$

computing their likelihoods, $\alpha_1^{(p)} = p(\boldsymbol{s}_1 \mid \boldsymbol{A}, \boldsymbol{W}^{(p)}[1], \{\boldsymbol{\theta}_{n\to n'}\}, \{\psi_n^{(0)}\})$, and initializing the particle weights to $v_1^{(p)} = \alpha_1^{(p)}$. Then, we iteratively proceed, updating the synaptic weights according to the learning rule and updating the particle weights according to the likelihood of the spikes.

That is, for $t = 2, \ldots, T$, we perform the following steps:

1. Sample the next synaptic weight given the weight in the preceding time bin, the learning rule, the spike history, and the global parameters,

$$\boldsymbol{W}^{(p)}[t] \sim p(\boldsymbol{W}[t] \mid \boldsymbol{W}^{(p)}[t-1], \boldsymbol{S}_{\leq t-1}, \boldsymbol{\vartheta}).$$

2. Compute the likelihood of the current spikes,

$$\alpha_t^{(p)} = p(\boldsymbol{s}_t \mid \boldsymbol{A}, \boldsymbol{W}^{(p)}[t], \{\boldsymbol{\theta}_{n\to n'}\}, \{\psi_n^{(0)}\}, \boldsymbol{S}_{\leq t})$$

3. Update the particle weight according to the likelihood of the current spikes,

$$v_t^{(p)} \leftarrow v_{t-1}^{(p)} \cdot \alpha_t^{(p)}.$$

This is, in fact, a special case of *sequential importance sampling* where our synaptic weights are sampled from the model's learning rule.

The problem with this simple algorithm is that the weights will rapidly decay to zero as particles drift from the regions of high likelihood. To counteract this effect, we often introduce a fourth step in which we *resample* the particles with replacement according to their weights.

4. Sample new particle indices with replacement according to the current weights,

$$p' \sim \text{Discrete}\left(\left[\frac{v_t^{(1)}}{\sum_p v_t^{(p)}}, \ldots, \frac{v_t^{(P)}}{\sum_p v_t^{(p)}}\right]\right),$$

and then replace the weight trajectories with those of the new particle indices,

$$\boldsymbol{W}^{(p)}[1, \ldots, t] \leftarrow \boldsymbol{W}^{(p')}[1, \ldots, t].$$

5. Once we have resampled the particle indices, we can reset the weights.

$$v_t^{(p)} \leftarrow \frac{1}{P}.$$

This is called *sequential importance resampling*. At the end of T time steps we are left with a weighted set of synaptic weight trajectories that approximates the conditional distribution over synaptic weights for given global model parameters.

6.3.2 Collapsed Gibbs Sampling of $\boldsymbol{A}$ and $\boldsymbol{W}[t]$

The particle weights also provide an unbiased estimate of the marginal likelihood of entries in the adjacency matrix, $\boldsymbol{A}$, integrating out the corresponding weight trajectory. We have,

$$\begin{aligned} p(\boldsymbol{A} \mid \boldsymbol{S}, \{\boldsymbol{\theta}_{n\to n'}\}, \{\psi_n^{(0)}\}) &\propto \sum_{t=1}^{T} \int p(\boldsymbol{A}, \boldsymbol{W}[t] \mid \boldsymbol{S}, \{\boldsymbol{\theta}_{n\to n'}\}, \{\psi_n^{(0)}\})\, \mathrm{d}\boldsymbol{W}[t] \\ &\approx \left[\prod_{t=1}^{T} \sum_{p=1}^{P} v_t^{(p)} \alpha_t^{(p)}\right] p(\boldsymbol{A} \mid \{\boldsymbol{z}_n\}, \boldsymbol{\vartheta}). \end{aligned}$$

We can leverage this estimator in a particle marginal Metropolis-Hastings (Andrieu et al., 2010) update. First, we propose an update to $\boldsymbol{A}$, then we run a particle filter to estimate the marginal likelihood of $\boldsymbol{A}$, and accept or reject the proposal accordingly. By marginalizing out the weight trajectory, we are able to explore the space of adjacency matrices more efficiently.

Factored Learning Rules If the learning rule factors into independent updates for each $w_{n'\to n}[t]$, we can update each synapse's weight trajectory separately and reduce the particles

to one-dimensional trajectories. The STDP learning rules considered in this chapter all factor in this way.

6.3.3 Particle MCMC

Particle filtering only yields a distribution over weight trajectories and implicitly assumes that the other parameters have been specified. Particle MCMC provides a broader inference algorithm for both weights and static parameters. The idea is to interleave *conditional* particle filtering steps that sample the weight trajectory given the current model parameters and the previously sampled weights, with traditional Gibbs updates to sample the model parameters given the current weight trajectory. This combination leaves the stationary distribution of the Markov chain invariant and allows joint inference over weights and parameters.

In our implementation, we also make use of a pMCMC variant with ancestor sampling (Lindsten et al., 2012) that significantly improves convergence. Any distribution may be used to propagate the particles forward; using the learning rule is simply the easiest to implement and understand. We have omitted a number of details in this description; for a thorough overview of particle MCMC, the reader should consult (Andrieu et al., 2010; Lindsten et al., 2012).

6.4 Evaluation

We evaluated our technique with two types of synthetic data. First, we generated data from our model, with known ground-truth. Second, we used the well-known simulator NEURON to simulate driven, interconnected populations of neurons undergoing synaptic plasticity. For comparison, we show how the sparse, time-varying GLM compares to a standard GLM with a group LASSO prior on the impulse response coefficients for which we can perform efficient MAP estimation.

6.4.1 GLM-based Simulations

As a proof of concept, we study a single synapse undergoing a variety of synaptic plasticity rules and generating spikes according to a GLM. The neurons also have inhibitory self-connections to mimic refractory effects. We tested three synaptic plasticity mechanisms: a static synapse (i.e., no plasticity), the unbounded, additive STDP rule given by Equation 6.3, and the bounded, multiplicative STDP rule given by Equation 6.4. For each learning rule, we simulated 60 seconds of spiking activity at 1kHz temporal resolution, updating the synaptic weights every 1s. The baseline firing rates were normally distributed with mean 20Hz and standard deviation of 5Hz. Correlations in the spike

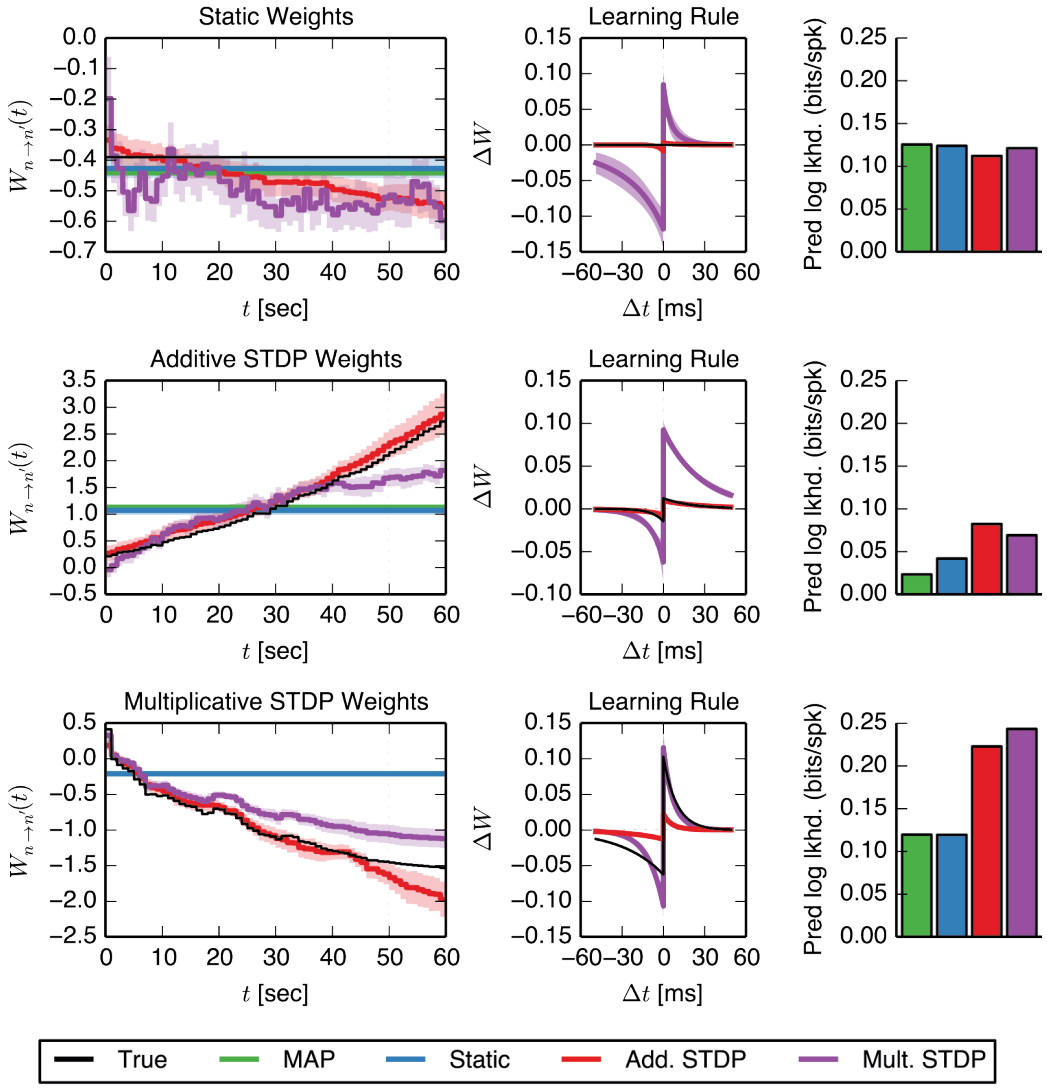

Figure 6.2: We fit time-varying weight trajectories to spike trains simulated from a GLM with two neurons undergoing no plasticity (top row), an additive, unbounded STDP rule (middle), and a multiplicative, saturating STDP rule (bottom row). We fit the first 50 seconds with four different models: MAP for an L1-regularized GLM, and fully-Bayesian inference for a static, additive STDP, and multiplicative STDP learning rules. In all cases, the correct models yield the highest predictive log likelihood on the final 10 seconds of the dataset.

timing led to changes in the synaptic weight trajectories that we could detect with our inference algorithm.

Figure 6.2 shows the true and inferred weight trajectories, the inferred learning rules, and the pre-

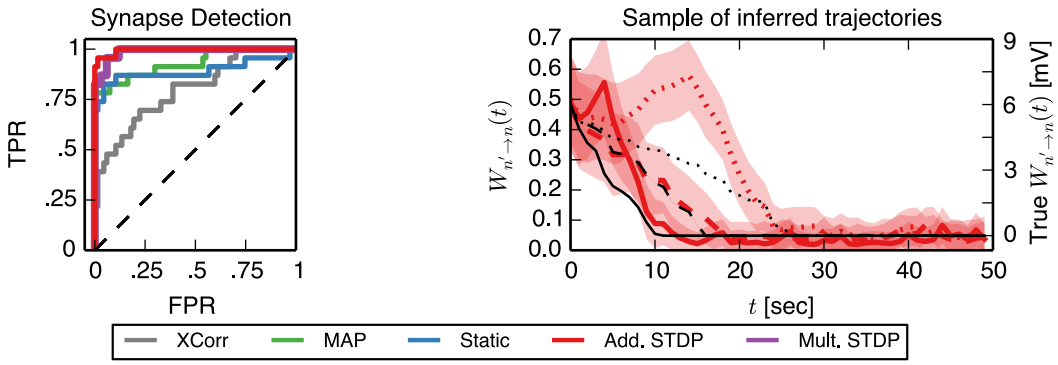

Figure 6.3: Evaluation of synapse detection on a 60 second spike train from a network of 10 neurons undergoing synaptic plasticity with a saturating, additive STDP rule, simulated with NEURON. The sparse, time-varying GLM with an additive rule outperforms the fully-Bayesian model with static weights, MAP estimation with L1 regularization, and simple thresholding of the cross-correlation matrix.

dictive log likelihood on ten seconds of held-out data for each of the three ground truth learning rules. When the underlying weights are static (top row), MAP estimation and static learning rules do an excellent job of detecting the true weight whereas the two time-varying models must compensate by either setting the learning rule as close to zero as possible, as the additive STDP does, or setting the threshold such that the weight trajectory is nearly constant, as the multiplicative model does. Note that the scales of the additive and multiplicative learning rules are not directly comparable since the weight updates in the multiplicative case are modulated by how close the weight is to the threshold. When the underlying weights vary (middle and bottom rows), the static models must compromise with an intermediate weight. Though the STDP models are both able to capture the qualitative trends, the correct model yields a better fit and better predictive power in both cases.

In terms of computational cost, our approach is clearly more expensive than alternative approaches based on MAP estimation or MLE. We developed a parallel implementation of our algorithm to capitalize on conditional independencies across neurons, i.e. for the additive and multiplicative STDP rules we can sample the weights $\boldsymbol{W}_{*\to n}$ independently of the weights $\boldsymbol{W}_{*\to n'}$. On the two neuron examples we achieve upward of 2 iterations per second (sampling all variables in the model), and we run our model for 1000 iterations. Convergence of the Markov chain is assessed by analyzing the log posterior of the samples, and typically stabilizes after a few hundred iterations. As we scale to networks of ten neurons, our running time quickly increases to roughly 20 seconds per iteration, which is mostly dominated by slice sampling the learning rule parameters. In order to evaluate the conditional probability of a learning rule parameter, we need to sample the weight trajectories for each synapse. Though these running times are nontrivial, they are not prohibitive for

networks that are realistically obtainable for optical study of synaptic plasticity.

6.4.2 Biophysical simulations

Using the biophysical simulator NEURON, we performed two experiments. First, we considered a network of 10 sparsely interconnected neurons (28 excitatory synapses) undergoing synaptic plasticity according to an additive STDP rule. Each neuron was driven independently by a hidden population of 13 excitatory neurons and 5 inhibitory neurons connected to the visible neuron with probability 0.8 and fixed synaptic weights averaging 3.0 mV. The visible synapses were initialized close to 6.0 mV and allowed to vary between 0.0 and 10.5 mV. The synaptic delay was fixed at 1.0 ms for all synapses. This yielded a mean firing rate of 10 Hz among visible neurons. Synaptic weights were recorded every 1.0 ms. These parameters were chosen to demonstrate interesting variations in synaptic strength, and as we transition to biological applications it will be necessary to evaluate the sensitivity of the model to these parameters and the appropriate regimes for the circuits under study.

We began by investigating whether the model is able to accurately identify synapses from spikes, or whether it is confounded by spurious correlations. Figure 6.3 shows that our approach identifies the 28 excitatory synapses in our network, as measured by ROC curve (Add. STDP AUC=0.99), and outperforms static models and cross-correlation. In the sparse, time-varying GLM, the probability of an edge is measured by the mean of $\boldsymbol{A}$ under the posterior, whereas in the standard GLM with MAP estimation, the likelihood of an edge is measured by area under the impulse response.

Looking into the synapses that are detected by the time-varying model and missed by the static model, we find an interesting pattern. The improved performance comes from synapses that decay in strength over the recording period. Three examples of these synaptic weight trajectories are shown in the right panel of Figure 6.3. The time-varying model assigns over 90% probability to each of the three synapses, whereas the static model infers less than a 40% probability for each synapse.

Finally, we investigated our model's ability to distinguish various learning rules by looking at a single synapse, analogous to the experiment performed on data from the GLM. Figure 6.4 shows the results of a weight trajectory for a synapse under additive STDP with a strict threshold on the excitatory post-synaptic current. The time-varying GLM with an additive model captures the same trajectory, as shown in the left panel. The GLM weights have been linearly rescaled to align with the true weights, which are measured in millivolts. Furthermore, the inferred additive STDP learning rule, in particular the time constants and relative amplitudes, perfectly match the true learning rule.

These results demonstrate that a sparse, time-varying GLM is capable of discovering synaptic weight trajectories, but in terms of predictive likelihood, we still have insufficient evidence to distin-

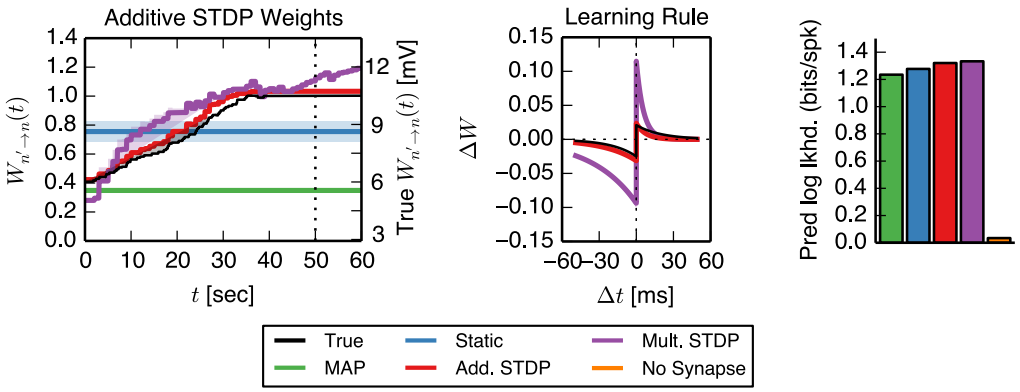

Figure 6.4: Analogously to Figure 6.2, a sparse, time-varying GLM can capture the weight trajectories and learning rules from spike trains simulated by NEURON. Here an excitatory synapse undergoes additive STDP with a hard upper bound on the excitatory post-synaptic current. The weight trajectory inferred by our model with the same parametric form of the learning rule matches almost exactly, whereas the static models must compromise in order to capture early and late stages of the data, and the multiplicative weight exhibits qualitatively different trajectories. Nevertheless, in terms of predictive log likelihood, we do not have enough information to correctly determine the underlying learning rule. Potential solutions are discussed in the main text.

guish additive and multiplicative STDP rules. By the end of the training period, the weights have saturated at a level that almost surely induces post-synaptic spikes. At this point, we cannot distinguish two learning rules which have both reached saturation. This motivates further studies that leverage this probabilistic model in an optimal experimental design framework, similar to recent work by Shababo et al. (2013), in order to conclusively test hypotheses about synaptic plasticity.

6.5 Discussion

Motivated by the advent of optical tools for interrogating networks of synaptically connected neurons, which make it possible to study synaptic plasticity in novel ways, we have extended the GLM to model a sparse, time-varying synaptic network, and introduced a fully-Bayesian inference algorithm built upon particle MCMC. Our initial results suggest that it is possible to infer weight trajectories for a variety of biologically plausible learning rules.

A number of interesting questions remain as we look to apply these methods to biological recordings. We have assumed access to precise spike times, though extracting spike times from optical recordings poses inferential challenges of its own. Solutions like those of Vogelstein et al. (2009) could be incorporated into our probabilistic model. Computationally, particle MCMC could be replaced with stochastic EM to achieve improved performance (Lindsten et al., 2012), and optimal

experimental design could aid in the exploration of stimuli to distinguish between learning rules. Beyond these direct extensions, this work opens up potential to infer latent state spaces with potentially nonlinear dynamics and non-Gaussian noise, and to infer learning rules at the synaptic or even the network level.

7

Bayesian Nonparametric Hidden Markov Models

The dynamic network models of the Chapter 6 introduced an important idea — the notion of a time-varying latent state. While we did not explicitly frame it in these terms, the dynamic weight matrices are an example of a latent population state. There, the state had a concrete biophysical interpretation and was imbued with dynamics derived from synaptic plasticity, but in general, latent state space models need not be tied to specific biophysical phenomena. In this chapter, we explore a very common state space model, the hidden Markov model (HMM), which can model neural spike trains as a progression of discrete latent states. By relating these states to externally measured covariates, we can gain insight into the computations performed by neural circuits.

Hidden Markov models have found numerous applications in neural modeling. Recently, Latimer et al. (2015) have used HMMs to argue that decision-making related activity in macaque lateral intraparietal (LIP) area is better characterized by a discrete step between "undecided" and "decided" states. Similarly, Miller and Katz (2010) have used HMMs to model the dynamics of neural activity during taste processing and decision making. In modeling the activity in hippocampal place cells, Chen et al. (2012; 2014) have used HMMs to interrogate discrete states of hippocampal activity and reconstruct topological maps of the environment from neural activity. This chapter provides further analysis of the same hippocampal dataset, which was also used in Chapters 3 and 5.

A major challenge with using HMMs in practice is determining the number of latent states. It is common to fit models of varying dimensionality and compare them on the basis of a cross-

validation metric, such as the predictive log-likelihood assigned to held-out data. However, this approach has a number of drawbacks. First, it can be computationally expensive to fit and compare a sequence of models. Second, it makes inefficient use of the data since we can only use a fraction of the data to train the model. While this is primarily a concern in the "small data" regime, it is still relevant to neural data analysis. Third, as with basic mixture models, the standard model does not explicitly penalize duplicate states. This penalty is only implicitly enforced through cross-validation. In terms of interpretability, it is desirable to incorporate an explicit penalty on excess states.

Here we extend the preceding work and consider a *Bayesian nonparametric* approach (Orbanz and Teh, 2011). The Bayesian nonparametric approach brings additional flexibility to the probabilistic model (Teh and Jordan, 2010; Wood and Black, 2008; Shalchyan and Farina, 2014). Specifically, we use a hierarchical Dirichlet process hidden Markov model (HDP-HMM) (Teh et al., 2006), which extends the finite-state HMM with an HDP prior that theoretically allows a countably infinite number of states. The Dirichlet process provides a nonparametric prior over atomic probability measures with support for a countably infinite number of outcomes. This is a natural way to model distributions over infinitely many states, e.g. to model the rows of a transition matrix. The *hierarchical* part of the HDP prior allows sharing between the rows. Even though these priors support infinitely many states, we can still perform efficient inference by leveraging truncation methods and constructive definitions of the HDP that only instantiate variables for states that are visited in the data.

Nevertheless, inference in Bayesian nonparametric models can be finicky. Hyperparameters of the HDP prior can have a strong effect, as can the particular choice of inference algorithm. Here we provide an assessment of various inference approaches. We consider both MCMC and variational inference algorithms. For MCMC, we adapt the Gibbs sampling approach of (Teh et al., 2006), and consider various methods of inferring hyperparameters.

We test the statistical model and inference methods with both simulation data and experimental data. The latter consists of a recording of rat dorsal hippocampal ensemble spike activity during open field navigation. Using a decoding analysis and predictive likelihood, we verify and compare the performance of the proposed Bayesian inference algorithms.

7.1 Probabilistic Model

First we present a standard Bayesian hidden Markov model with a fixed, finite number of states. We introduce notation and prior distributions for the various model parameters. Then we extend this

to the nonparametric case with a countably infinite number of latent states.

7.1.1 Parametric Hidden Markov Models

Consider a finite K-state HMM applied to population spiking activity from a population of N neurons. We assume that the discrete latent state follows a first-order Markov chain $\boldsymbol{z} = [z_1, \ldots, z_T]$ with $z_t \in \{1, \ldots, K\}$, and that the spike counts of individual neurons at time t follow a Poisson distribution whose rate depends on the latent state, z_t. This is summarized in the following probabilistic model:

$$p(\boldsymbol{S}, \boldsymbol{z} \mid \boldsymbol{\pi}^{(0)}, \boldsymbol{P}, \boldsymbol{\Lambda}) = p(z_1 \mid \boldsymbol{\pi}) \prod_{t=2}^{T} p(z_t \mid z_{t-1}, \boldsymbol{P}) \prod_{t=1}^{T} p(\boldsymbol{s}_t \mid z_t, \boldsymbol{\Lambda}), \tag{7.1}$$

where

$$\begin{aligned} p(z_1 \mid \boldsymbol{\pi}^{(0)}) &= \mathrm{Discrete}(z_1 \mid \boldsymbol{\pi}^{(0)}), \\ p(z_t \mid z_{t-1}, \boldsymbol{P}) &= \mathrm{Discrete}(z_t \mid \boldsymbol{\pi}^{(z_{t-1})}), \\ p(\boldsymbol{s}_t \mid z_t, \boldsymbol{\Lambda}) &= \prod_{n=1}^{N} \mathrm{Poisson}(s_{t,n} \mid \lambda_{z_t,n}). \end{aligned}$$

Here, $\boldsymbol{\pi}^{(0)} \in [0,1]^K$ is a discrete probability distribution over initial states, and

$$\boldsymbol{P} = \begin{bmatrix} — & \boldsymbol{\pi}^{(1)} & — \\ & \vdots & \\ — & \boldsymbol{\pi}^{(K)} & — \end{bmatrix},$$

is a $K \times K$ transition matrix where the row, $\boldsymbol{\pi}^{(k)} \in [0,1]^K$, specifies a discrete conditional distribution over z_t given that $z_{t-1} = k$. The state-conditional firing rates are collected in the matrix,

$$\boldsymbol{\Lambda} = \begin{bmatrix} — & \boldsymbol{\lambda}^{(1)} & — \\ & \vdots & \\ — & \boldsymbol{\lambda}^{(K)} & — \end{bmatrix} = \begin{bmatrix} \lambda_{1,1} & \ldots & \lambda_{1,N} \\ \vdots & & \vdots \\ \lambda_{K,1} & \ldots & \lambda_{K,N} \end{bmatrix},$$

In a Bayesian HMM, we introduce prior distributions over the parameters. We use the following

prior distributions,

$$\begin{aligned}
\alpha_0 &\sim \text{Gamma}(a_{\alpha_0}, 1.0) \\
\boldsymbol{\pi}^{(0)} &\sim \text{Dir}(\alpha_0 \mathbf{1}), \\
\boldsymbol{\pi}^{(k)} &\sim \text{Dir}(\alpha_0 \mathbf{1}), \\
\lambda_{k,n} &\sim \text{Gamma}(\kappa_n, \nu_n).
\end{aligned}$$

where gamma prior on firing rates has neuron-specific shape parameters, κ_n, and scale parameters, ν_n.

7.1.2 Nonparametric Hidden Markov Models

Model selection is an important issue for statistical modeling and data analysis. Here we extend the finite-state HMM to an HDP-HMM: a Bayesian nonparametric extension of the HMM that allows for a potentially infinite number of hidden states (Teh et al., 2006; Beal et al., 2002). The HDP-HMM treats the priors via a stochastic process. Instead of imposing a Dirichlet prior distribution on the rows of the finite state transition matrix $\boldsymbol{P}$, we use a HDP that allows for a countably infinite number of states.

Specifically, we sample a distribution over latent states, G_0, from a Dirichlet process (DP) (Ferguson, 1973) prior, $G_0 \sim \mathcal{DP}(\gamma, H)$, where γ is the concentration parameter and H is the base measure. Moreover, we place a prior distribution over the concentration parameter, $\gamma \sim \text{Gamma}(a_\gamma, 1.0)$. Given the concentration, one may sample from the DP via the "stick-breaking construction" (Sethuraman, 1994). First, sample the stick-breaking weights, $\boldsymbol{\beta}$,

$$\widetilde{\beta}_k \sim \text{Beta}(1, \gamma), \quad \beta_k = \widetilde{\beta}_k \prod_{j=1}^{k-1} \left(1 - \widetilde{\beta}_j\right), \tag{7.2}$$

where $\beta_1 = \widetilde{\beta}_1$, $\sum_{k=1}^{\infty} \beta_k = 1$.

The stick-breaking construction of (7.2) is sometimes denoted as $\boldsymbol{\beta} \sim \text{GEM}(\gamma)$, after Griffiths, Engen, and McCloskey (Ewens, 1990). The name "stick-breaking" comes from the interpretation of β_k as the length of the piece of a unit-length stick assigned to the k-th value. After the first $k-1$ values having their portions assigned, the length of the remainder of the stick is broken according to a sample $\widetilde{\beta}_k$ from a beta distribution, and β_k indicates the portion of the remainder to be assigned to the k-th value. Therefore, the stick-breaking process $\text{GEM}(\gamma)$ also defines a DP— smaller values

of γ will lead to larger values of $\widetilde{\beta}_k$, which means most of the probability mass will be allocated to the first "sticks," i.e. the small values of k.

After sampling $\boldsymbol{\beta}$, we next sample the latent state variables, in this case $\boldsymbol{\lambda}^{(k)}$, from the base measure H. For us, H is simply a set of independent gamma distributions for each neuron. Our draw from the $\mathcal{DP}(\gamma, H)$ prior is then given by

$$G_0(\boldsymbol{\lambda}) = \sum_{k=1}^{\infty} \beta_k \, \delta_{\boldsymbol{\lambda}^{(k)}}(\boldsymbol{\lambda}).$$

Thus, the stick breaking construction makes clear that draws from a Dirichlet process distribution are discrete with probability one.

Given a countably infinite set of shared states, we may then sample the rows of the transition matrix, $\boldsymbol{\pi}^{(k)} \sim \mathcal{DP}(\alpha_0, \boldsymbol{\beta})$. We place the same prior over $\boldsymbol{\pi}^{(0)}$. The base measure in this case is $\boldsymbol{\beta}$, a countably infinite vector of stick-breaking weights, that serves as the mean of the DP prior over the rows of $\boldsymbol{P}$. The concentration parameter, α_0, governs how concentrated the rows are about the mean. Since the base measure $\boldsymbol{\beta}$ is discrete, each row of $\boldsymbol{P}$ will be able to "see" the same set of states. By contrast, if we remove the HDP prior and treat each row of $\boldsymbol{P}$ as an independent draw from a DP with base measure H, each row would see a disjoint set of states with probability one. In other words, the hierarchical prior is required to provide a discrete (but countably infinite) set of latent states for the HMM.

7.2 Markov Chain Monte Carlo Inference

Several MCMC-based inference methods have been developed for the HDP-HMM (Teh et al., 2006; Van Gael et al., 2008). Some of these previous works use a collapsed Gibbs sampler in which the transition matrix $\boldsymbol{P}$ and the observation parameters $\boldsymbol{\Lambda}$ are integrated out (Teh et al., 2006; Van Gael et al., 2008). In this work, however, we use a "weak limit" approximation in which the

DP prior is approximated with a symmetric Dirichlet prior. Specifically, we let

$$\begin{aligned}
\gamma &\sim \text{Gamma}(a_\gamma, 1), \\
\alpha_0 &\sim \text{Gamma}(a_{\alpha_0}, 1), \\
\boldsymbol{\beta} \,|\, \gamma &\sim \text{Dir}(\gamma/K_{\text{max}}, \ldots, \gamma/K_{\text{max}}), \\
\boldsymbol{\pi}^{(0)} \,|\, \alpha_0, \boldsymbol{\beta} &\sim \text{Dir}(\alpha_0\beta_1, \ldots, \alpha_0\beta_{K_{\text{max}}}), \\
\boldsymbol{\pi}^{(k)} \,|\, \alpha_0, \boldsymbol{\beta} &\sim \text{Dir}(\alpha_0\beta_1, \ldots, \alpha_0\beta_{K_{\text{max}}}).
\end{aligned} \tag{7.3}$$

where K_{max} denotes a truncation level. It can be shown that this prior will weakly converge to the DP prior as the dimensionality of the Dirichlet distribution approaches infinity (Johnson and Willsky, 2014; Ishwaran and Zarepour, 2002). With this approximation we can capitalize on forward-backward sampling algorithms to jointly update the latent states $\boldsymbol{z}$.

Previous work has typically been presented with Gaussian or multinomial likelihood models, with the acknowledgment that the same methods work with any exponential family likelihood when the base measure H is a conjugate prior. Here we present the Gibbs sampling algorithm of (Teh et al., 2006) for the HDP-HMM applied to the special case of independent Poisson observations, and we derive Hamiltonian Monte Carlo (HMC) (Neal, 2010) transitions to sample the neuron-specific hyperparameters of the firing rate priors.

We begin by defining Gibbs updates for the neuronal firing rates $\boldsymbol{\Lambda}$. Since we are using gamma priors with independent Poisson observations, the model is fully conjugate and simple Gibbs updates suffice. Therefore, we have

$$\lambda_{k,n} \,|\, \boldsymbol{S}, \boldsymbol{z} \sim \text{Gamma}\left(\kappa_n + \sum_{t=1}^{T} s_{t,n}\, \mathbb{I}[z_t = k],\ \nu_n + \sum_{t=1}^{T} \mathbb{I}[z_t = k]\right).$$

Under the weak limit approximation the priors on $\boldsymbol{\pi}^{(k)}$ and $\boldsymbol{\pi}^{(0)}$ reduce to Dirichlet distributions, which are also conjugate with the finite HMM. Hence we can derive conjugate Gibbs updates

for these parameters as well. They take the form:

$$\begin{aligned}
\boldsymbol{\pi}^{(0)} \,|\, \alpha_0, \boldsymbol{\beta} &\sim \mathrm{Dir}\left(\alpha_0 \boldsymbol{\beta} + \mathbf{1}_{z_1}\right), \\
\boldsymbol{\pi}^{(k)} \,|\, \alpha_0, \boldsymbol{\beta} &\sim \mathrm{Dir}\left(\alpha_0 \boldsymbol{\beta} + \boldsymbol{n}_k\right), \\
n_{i,j} &= \sum_{t=1}^{T-1} \mathbb{I}[z_t = i] \cdot \mathbb{I}[z_{t+1} = j],
\end{aligned}$$

where $\mathbf{1}_k$ is a unit vector with a one in the k-th entry.

The Dirichlet parameters $\boldsymbol{\beta}$ and the concentration parameters α_0 and γ can be updated as in (Teh et al., 2006).

7.2.1 Block Gibbs updates for the latent states

Conditioned upon the firing rates, the initial state distribution, and the transition matrix, which we collectively refer to as $\boldsymbol{\theta}$, we can jointly update the latent states of the HDP-HMM using a *forward filtering, backward sampling* algorithm. Jointly sampling these latent states allows us to avoid issues with mixing when individually sampling states that are highly correlated with one another. We provide a brief overview of this algorithm here. Complete details of this algorithm can be found in, for example, Johnson (2014).

First, we "filter" the data to get the marginal distribution over z_t given the observations up to time t. We use "Matlab" notation to refer to a set of variables, $\boldsymbol{s}_{1:t} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t\}$. Since z_t is discrete, its filtered distribution is parameterized by a probability vector, which we call $\boldsymbol{m}_t$.

We compute these filtered probability distributions iteratively. Assume that at iteration t we have already computed $\boldsymbol{m}_{t-1}$. Given the Markovian structure of the probabilistic model, the conditional distribution of z_t factors into,

$$p(z_t \,|\, \boldsymbol{s}_{1:t}, \boldsymbol{\theta}) \propto \underbrace{p(\boldsymbol{s}_t \,|\, z_t, \boldsymbol{\theta})}_{\text{condition}} \underbrace{p(z_t \,|\, \boldsymbol{s}_{1:t-1}, \boldsymbol{\theta})}_{\text{predict}}.$$

The *prediction* step involves a marginalization over the previous latent state, z_{t-1},

$$\begin{aligned}
p(z_t \,|\, \boldsymbol{s}_{1:t-1} \boldsymbol{\theta}) &\propto \sum_{k=1}^{K} p(z_t \,|\, z_{t-1} = k, \boldsymbol{\theta})\, p(z_{t-1} = k \,|\, \boldsymbol{s}_{1:t-1}, \boldsymbol{\theta}) \\
&= \mathrm{Discrete}(z_t \,|\, \boldsymbol{m}_{t|t-1}),
\end{aligned}$$

where

$$m_{t|t-1,k} \propto \sum_{j=1}^{K} p(z_t = k \mid z_{t-1} = j, \boldsymbol{\theta}) \cdot m_{t-1,j}.$$

Then, we *condition* on the current observations, $\boldsymbol{s}_t$, to get the parameters of the filtered distribution,

$$m_{t,k} \propto p(\boldsymbol{s}_t \mid z_t = k, \boldsymbol{\theta}) \cdot m_{t|t-1,k}. \tag{7.4}$$

Once we have computed the filtered distributions for all time bins, we can sample from the joint distribution over $\boldsymbol{z}_{1:T}$ by applying the chain rule,

$$\begin{aligned} p(\boldsymbol{z}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{\theta}) &= p(z_T \mid \boldsymbol{s}_{1:T}, \boldsymbol{\theta}) \prod_t p(z_t \mid \boldsymbol{z}_{t+1:T}, \boldsymbol{s}_{1:T}, \boldsymbol{\theta}) \\ &\propto p(z_T \mid \boldsymbol{s}_{1:T}, \boldsymbol{\theta}) \prod_t p(z_t \mid \boldsymbol{s}_{1:t}, \boldsymbol{\theta})\, p(z_{t+1} \mid z_t, \boldsymbol{\theta}). \end{aligned}$$

Thus, we can sample in reverse order, starting with z_T and ending with z_1. The conditional distribution of z_t is,

$$p(z_t \mid \boldsymbol{z}_{t+1:T}, \boldsymbol{s}_{1:T}, \boldsymbol{\theta}) \propto p(z_t \mid \boldsymbol{m}_t)\, p(z_{t+1} \mid z_t, \boldsymbol{\theta}), \tag{7.5}$$

which is another discrete distribution. The final algorithm for block Gibbs sampling $\boldsymbol{z}_{1:T}$ is:

Require: $\boldsymbol{s}_{1:T}, \boldsymbol{\theta}$
  for $t = 1, \ldots, T$ **do**
    Compute $\boldsymbol{m}_t$ ▷ Eq. 7.4
  end for
  for $t = T, \ldots, 1$ **do**
    Sample $z_t \mid z_{t+1}, \boldsymbol{m}_t, \boldsymbol{\theta}$ ▷ Eq. 7.5
  end for

Algorithm 7.1: Forward filtering, backward sampling algorithm for the hidden Markov model.

A single iteration of the complete Gibbs sampling algorithm consists of an update for each parameter of the model. The aforementioned updates are based upon previous work; one novel direction that we explore in this chapter is the sampling of the hyperparameters of the gamma firing rate priors.

7.2.2 Setting firing rate hyperparameters

We consider three approaches to setting the hyperparameters of the gamma priors for Poisson firing rates, namely, $\{\kappa_n, \nu_n\}$ for the n-th neuron.

- In the first approach, we estimate these parameters using an empirical Bayesian (EB) procedure, that is, by maximizing the marginal likelihood of the spike counts. For each neuron, this may be easily done using standard maximum likelihood estimation for the negative binomial model. In practice, we found that without regularization this approach leads to extreme values of the hyperparameters.

- Our second approach samples these hyperparameters using Hamiltonian Monte Carlo (HMC) (Neal, 2010). We note that for fixed values of the "shape" parameter κ_n, the conditional distribution of the "scale" parameter, ν_n is conjugate with a gamma prior distribution. However, setting the shape parameter *a priori* is challenging because it can have a strong influence on the firing rate distribution. HMC allows us to jointly sample both the shape and the scale parameters simultaneously.

  To implement HMC we must have access to both the log probability of the parameters as well as its gradient. Since both parameters are restricted to be positive, we instead reparameterize the problem in terms of their logs. For neuron n, the conditional log probability equal to,

$$\begin{aligned}
\mathcal{L} &= \log p(\log \kappa_n, \log \nu_n \mid \mathbf{\Lambda}) \\
&= \sum_{k=1}^{K} \log p(\lambda_{k,n} \mid \kappa_n, \nu_n) + \text{const.} \\
&= \sum_{k=1}^{K} \kappa_n \log \nu_n - \log \Gamma(\kappa_n) + (\kappa_n - 1) \log \lambda_{k,n} - \nu_n \lambda_{k,n}.
\end{aligned}$$

  Taking gradients with respect to both parameters yields,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \log \kappa_n} &= \sum_{k=1}^{K} \left[\log \nu_n - \Psi(\kappa_n) + \log \lambda_{k,n}\right] \times \kappa_n, \\
\frac{\partial \mathcal{L}}{\partial \log \nu_n} &= \sum_{k=1}^{K} \left[\frac{\kappa_n}{\nu_n} - \lambda_{k,n}\right] \times \nu_n.
\end{aligned}$$

The HMC algorithm uses these gradients to inform a stochastic walk over the posterior distribution. With knowledge of the gradients, HMC can sometimes make large updates to parameters, especially in cases where the parameters are highly correlated under the posterior.

- In the final approach, we fix the shape hyperparameter, κ_n, and infer the scale, ν_n. We place a gamma prior on the scale, $\nu_n \sim \mathrm{Gamma}(\mu, \nu_0)$. Given κ_n, the conditional distribution of the scale is

$$\nu_n \,|\, \kappa_n, \{\lambda_{k,n}\}, \boldsymbol{z} \sim \mathrm{Gamma}\Big(\mu + \sum_{k=1}^{K} \mathbb{I}[n_k > 0] \cdot \kappa_n,\ \nu_0 + \sum_{k=1}^{K} \mathbb{I}[n_k > 0] \cdot \lambda_{k,n}\Big)$$
$$n_k = \sum_{t=1}^{T} \mathbb{I}[z_t = k].$$

  In the following experiments, we set the shape parameter be $\kappa_n = 1$, and we set the scale prior parameters to $\mu = 1$ and $\nu = 1$. This is equivalent to an exponential prior on rates, $\lambda_{k,n} \sim \mathrm{Exp}(\nu_n)$, and an exponential prior on the scale $\nu_n \sim \mathrm{Exp}(1)$. One could perform cross validation over the shape parameter, but the exponential prior is a rather weak assumption that enables fully-Bayesian inference.

7.2.3 Predictive log likelihood

With the parameter and hyperparameter inference complete, we evaluate the performance of our algorithm in terms of its predictive log likelihood on held-out test data. We approximate the predictive log likelihood with samples from the posterior distribution generated by our MCMC algorithm. That is,

$$\begin{aligned}\log p(\boldsymbol{S}_{\mathsf{test}} \,|\, \boldsymbol{S}_{1:T}) &= \log \sum_{\boldsymbol{z}_{\mathsf{test}}} \int_{\boldsymbol{\Theta}} p\left(\boldsymbol{S}_{\mathsf{test}}, \boldsymbol{z}_{\mathsf{test}} \,|\, \boldsymbol{\theta}\right)\ p\left(\boldsymbol{\theta} \,|\, \boldsymbol{S}_{\mathsf{train}}\right) \mathrm{d}\boldsymbol{\theta}, \\ &\approx \log \frac{1}{L} \sum_{\ell=1}^{L} \sum_{\boldsymbol{z}_{\mathsf{test}}} p\left(\boldsymbol{S}_{\mathsf{test}}, \boldsymbol{z}_{\mathsf{test}} \,|\, \boldsymbol{\theta}^{(\ell)}\right),\end{aligned}$$

where $\boldsymbol{\theta} = (\boldsymbol{\Lambda}, \boldsymbol{P}, \boldsymbol{\pi}^{(0)})$ and $\{\boldsymbol{\theta}^{(\ell)}\}_{\ell=1}^{L} \sim p(\boldsymbol{\theta} \,|\, \boldsymbol{S}_{\mathsf{train}})$. The summation over latent state sequences for the test data is performed with the message-passing algorithm for HMMs.

7.3 Variational Inference

We build upon our previous work (Chen et al., 2012; 2014; Johnson and Willsky, 2014) to develop a variational inference algorithm for fitting the HDP-HMM to hippocampal spike trains. Our objective is to approximate the posterior distribution of the HDP-HMM with a distribution from a more tractable family. As usual, we choose a factorized approximation that allows for tractable optimization of the parameters of the variational model. Specifically, we let,

$$p(\boldsymbol{z}, \boldsymbol{\Lambda}, \boldsymbol{P}, \boldsymbol{\pi}^{(0)}, \boldsymbol{\beta} \mid \boldsymbol{S}_{1:T}) \approx q(\boldsymbol{z})\, q(\boldsymbol{\Lambda})\, q(\boldsymbol{P})\, q(\boldsymbol{\pi}^{(0)})\, q(\boldsymbol{\beta}).$$

Since the independent Poisson observations are conjugate with the gamma firing rate prior distributions, choosing a set of independent gamma distributions for $q(\boldsymbol{\Lambda})$ allows for simple variational updates.

$$\begin{aligned}
q(\boldsymbol{\Lambda}) &= \prod_{k=1}^{K} \prod_{n=1}^{N} \text{Gamma}\left(\widetilde{\kappa}_{k,n},\, \widetilde{\nu}_{k,n}\right), \\
\widetilde{\kappa}_{k,n} &\leftarrow \kappa_n + \sum_{t=1}^{T} s_{t,n} \mathbb{E}_q\big[\mathbb{I}[z_t = k]\big], \\
\widetilde{\nu}_{k,n} &\leftarrow \nu_n + \sum_{t=1}^{T} \mathbb{E}_q\big[\mathbb{I}[z_t = k]\big].
\end{aligned}$$

Following (Johnson and Willsky, 2014), we use a "direct assignment" truncation for the HDP (Bryant and Sudderth, 2012; Liang et al., 2007). In this scheme, a truncation level K_{max} is chosen *a priori* and $q(\boldsymbol{z})$ is limited to support only states $z_t \in \{1, \ldots, K_{\mathsf{max}}\}$. The advantage of this approximation is that conjugacy is retained with $\boldsymbol{\Lambda}$, $\boldsymbol{P}$, and $\boldsymbol{\pi}^{(0)}$, and the variational approximation $q(\boldsymbol{z})$ reduces to*

$$\begin{aligned}
q(\boldsymbol{z}) &= \text{HMM}(\widetilde{\boldsymbol{P}}, \widetilde{\boldsymbol{\pi}}^{(0)}, \widetilde{\boldsymbol{\Lambda}}), \\
\widetilde{\boldsymbol{P}} &= \exp\left\{\mathbb{E}_q[\ln \boldsymbol{P}]\right\}, \\
\widetilde{\boldsymbol{\pi}}^{(0)} &= \exp\left\{\mathbb{E}_q[\ln \boldsymbol{\pi}^{(0)}]\right\}, \\
\widetilde{\boldsymbol{\Lambda}} &= \exp\left\{\mathbb{E}_q[\ln p(\boldsymbol{S} \mid \boldsymbol{\Lambda})]\right\}.
\end{aligned}$$

*In a slight abuse of notation, $\widetilde{\boldsymbol{\Lambda}}$ refers to the expected observation likelihood for each latent state. That is, $\widetilde{\boldsymbol{\Lambda}}$ is a matrix where $\widetilde{\Lambda}_{t,k} = \exp\{\mathbb{E}_{q(\boldsymbol{\Lambda})}[\ln p(\boldsymbol{s}_t \mid z_t = k, \boldsymbol{\Lambda}]\}$.

Expectations $\mathbb{E}_q[z_t = k]$ can then be computed using standard message-passing algorithms for HMMs.

With the direct assignment truncation, the variational factors for the rows $\boldsymbol{\pi}^{(k)}$ and the initial distribution $\boldsymbol{\pi}^{(0)}$ are Dirichlet distributions. Unlike in the finite-state HMM, however, these Dirichlet factors are now over $K_{\mathsf{max}} + 1$ dimensions since the final dimension accounts for all states $k > K_{\mathsf{max}}$. Under the HDP prior we had $\boldsymbol{\pi}^{(k)} \sim \mathcal{DP}(\alpha_0 \cdot \boldsymbol{\beta})$, and under the truncation the DP parameter becomes $\alpha_0 \cdot \boldsymbol{\beta}_{1:K_{\mathsf{max}}+1}$. Again, leveraging the conjugacy of the model, we arrive at the following variational updates:

$$
\begin{aligned}
q(\boldsymbol{P}) &= \prod_{k=1}^{K_{\mathsf{max}}} \mathrm{Dir}(\widetilde{\boldsymbol{n}}_k), \\
\widetilde{n}_{i,j} &\leftarrow \alpha_0 \beta_j + \mathbb{E}_q[\mathbb{I}[z_t = i] \cdot \mathbb{I}[z_{t+1} = j]].
\end{aligned}
$$

We use an analogous update for $\boldsymbol{\pi}^{(0)}$.

The principal drawback of the direct assignment truncation is that the prior for $\boldsymbol{\beta}$ is no longer conjugate. This could be avoided with the fully conjugate approach of (Hoffman et al., 2013), however, this results in extra bookkeeping and the duplication of states. Instead, following (Johnson and Willsky, 2014; Bryant and Sudderth, 2012; Liang et al., 2007), we use a point estimate for this parameter by setting $q(\boldsymbol{\beta}) = \delta_{\boldsymbol{\beta}^*}(\boldsymbol{\beta})$ and use gradient ascent to update this parameter during inference.

There are a number of hyperparameters to set for the variational approach as well. The hyperparameters κ_n and ν_n of gamma prior on firing rates can be set with empirical Bayes, as above. We resort to cross validation to set the Dirichlet parameter α_0 and the GEM parameter γ.

7.3.1 Predictive Log Likelihood

Finally, in order to compute predictive log likelihoods on held-out test data, we draw multiple samples, $\{\boldsymbol{\theta}^{(\ell)}\}_{\ell=1}^{L}$ for $\boldsymbol{\theta} = (\boldsymbol{\Lambda}, \boldsymbol{z}, \boldsymbol{P}, \boldsymbol{\pi}^{(0)}, \boldsymbol{\beta})$, from the variational posterior, q, and approximate the predictive log likelihood as

$$
\begin{aligned}
\ln p(\boldsymbol{S}_{\mathsf{test}} \mid \boldsymbol{S}) &\approx \ln \mathbb{E}_q \left[p(\boldsymbol{S}_{\mathsf{test}} \mid \boldsymbol{\theta}) \right] \\
&\approx \ln \frac{1}{L} \sum_{\ell=1}^{L} p(\boldsymbol{S}_{\mathsf{test}} \mid \boldsymbol{\theta}^{(\ell)}).
\end{aligned}
$$

The inference algorithms were implemented based upon the PyHSMM framework of (John-

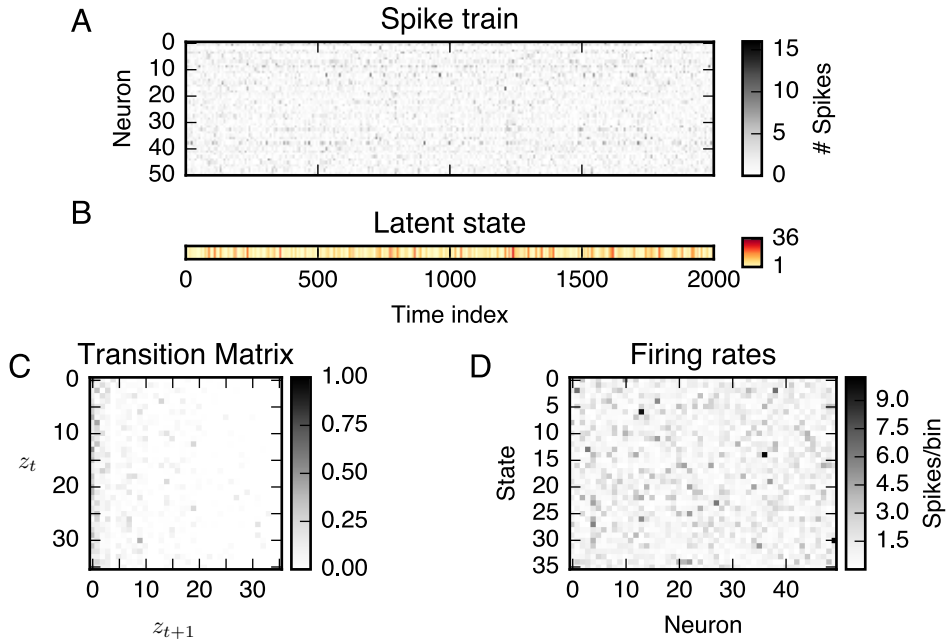

Figure 7.1: An example of a synthetic dataset drawn from an HDP-HMM. (A) Simulated population spike trains or spike counts. (B) Inferred latent state sequence. (C) Inferred state transition matrix $\boldsymbol{P}$. (D) Inferred neuronal firing rate matrix, $\boldsymbol{\Lambda}$.

son, 2014). The code-base was written in Python with C offloads for the message passing algorithms. We have extended the code-base to perform hyperparameter inference using the methods described above, and expanded it to tailor to neural spike train analysis. Our code is publicly available (`https://github.com/slinderman/pyhsmm_spiketrains`).

7.4 Synthetic Data Experiments

Setup First, we simulate synthetic spike count data using an HDP-HMM with $N = 50$ neurons, $T = 2000$ time bins, and Dirichlet concentration parameters $\alpha_0 = 12.0$ and $\gamma = 12.0$. These configuration yield state sequences that tend to visit 30–45 states. All of neuronal firing rate parameters are drawn from a gamma distribution: $\mathrm{Gamma}(\kappa_n = 1, \nu_n = 1)$ (with mean 1.0 and standard deviation 1.0).

An example of one such synthetic dataset is shown in Fig. 7.1. The states have been ordered according to their occupancy (i.e., how many times they are visited during the simulation), such that the columns of the transition matrix exhibit a decrease in probability as the incoming state num-

ber, z_{t+1}, increases. This is a characteristic of the HDP-HMM, indicating the tendency of the model to reuse states with high occupancy.

We compare six combinations of model, inference algorithm, and hyperparameter selection approaches: (i) HMM with the correct number of states, fit by Gibbs sampling with fixed $\kappa_n = 1$; (ii) HMM with the correct number of states, fit by VB with hyperparameters set by empirical Bayes; (iii) HDP-HMM fit by Gibbs sampling with fixed $\kappa_n = 1$; (iv) HDP-HMM fit by Gibbs sampling and HMC for hyperparameter updates; (v) HDP-HMM fit by MCMC with hyperparameters set by empirical Bayes; and (vi) HDP-HMM fit by VB with hyperparameters set by empirical Bayes. For the MCMC methods, we set gamma priors over the concentration parameters (α_0 and γ); for the VB methods, we set α_0 and γ to their true values. Alternatively, they can be selected by cross validation. We set both the weak limit approximation for MCMC and the direct assignment truncation level for VB to $K_{\mathsf{max}} = 100$.

We collect 5000 samples from the MCMC algorithms and use the last 2000 for computing predictive log likelihoods. For visualization, we use the final sample to extract the transition matrix and the firing rates. The number of samples and the amount of burn-in iterations were chosen by examining the log probability and parameter traces for convergence. It is found that the MCMC algorithm converges within hundreds of iterations. For further convergence diagnosis of a single Gibbs chain, one may use the autocorrelation tools suggested in (Raftery and Lewis, 1992; Cowles and Carlin, 1996).

We run the VB algorithm for 200 steps to guarantee convergence of the variational lower bound. Again, this is assessed by examining the variational lower bound and is found to converge to a local maxima within tens of iterations.

Assessment We use two criteria for result assessment with simulation data. The first criterion is based on the Hamming error between the true and inferred state sequences. To compute this, we first relabel the inferred states in order to maximize overlap with the true states. Let $\boldsymbol{z}$ be the true state sequence and $\boldsymbol{z}'$ be the inferred state sequence. We define the overlap matrix $O \in \mathbb{N}^{K_{\mathsf{max}} \times K_{\mathsf{max}}}$ whose entries $O_{i,j}$ is the number of times the true state is i and the inferred state is j:

$$O_{i,j} = \sum_{t=1}^{T} \mathbb{I}[z_t = i]\, \mathbb{I}[z'_t = j].$$

We use the Hungarian method (Kuhn, 1955) to find a relabeling of the inferred states that maximizes overlap, and then we measure the Hamming error between the true state sequence $\boldsymbol{z}$, and the rela-

Dataset	1	2	3	4	5
HMM (Gibbs)	9	401	13	24	615
HMM (VB)	166	290	295	123	124
HDP-HMM (Gibbs)	2	**3**	5	**1**	6
HDP-HMM (HMC)	3	4	3	2	**4**
HDP-HMM (EB)	**1**	**3**	**2**	3	12
HDP-HMM (VB)	432	586	340	264	675

Table 7.1: Comparison of Hamming error (see Eq. 7.6) computed from the same nine simulated data sets as above. The VB inference methods tend to overestimate the number of states and therefore have much higher Hamming error.

beled sequence of inferred states, $\widetilde{\boldsymbol{z}}'$:

$$\mathrm{err}(\boldsymbol{z}, \widetilde{\boldsymbol{z}}') = \sum_{t=1}^{T} \mathbb{I}[z_t \neq \widetilde{z}'_t]. \tag{7.6}$$

Table 7.1 summarizes the Hamming error for all six models on five synthetic datasets. We see that the HDP-HMM fit via Gibbs sampling with firing rate hyperparameters set via empirical Bayes outperforms the other models and inference algorithms on three of five datasets, but the HDP-HMM with hyperparameter HMC sampling are very comparable. By contrast, when the models are fit with VB inference, the inferred state sequences tend to use more than the true number of states, which results in very poor Hamming error. Similarly, the HMM fit via Gibbs sampling does not factor in the penalty on additional states and instead tends to use all states equally, resulting in high Hamming error.

The second criterion is the model's predictive log likelihood (bits/spike) on a held-out sequence of $T_{\mathsf{test}} = 1000$ time steps. We compare the predictive log likelihood to that of a set of independent Poisson processes. Their rates and the corresponding predictive log likelihood are given by,

$$\widehat{\lambda}_n = \frac{1}{T_{\mathsf{train}}} \sum_{t=1}^{T_{\mathsf{train}}} s_{t,n},$$

$$\log p(\boldsymbol{S}_{\mathsf{test}} \mid \boldsymbol{S}_{\mathsf{train}}) = \sum_{n=1}^{N} \left[-T_{\mathsf{test}} \widehat{\lambda}_n + \sum_{t=1}^{T_{\mathsf{test}}} s_{t,n} \log \widehat{\lambda}_n \right].$$

The improvement obtained by a model is measured in bits, and is normalized by the number of spikes in the test dataset in order to obtain comparable units for each of the test datasets.

Dataset	1	2	3	4	5
HMM (Gibbs)	0.315	0.300	0.312	0.310	0.250
HMM (VB)	0.298	0.290	0.313	0.306	0.252
HDP-HMM (Gibbs)	**0.323**	**0.307**	**0.321**	**0.318**	**0.259**
HDP-HMM (HMC)	**0.323**	0.306	0.320	**0.318**	**0.259**
HDP-HMM (EB)	0.322	0.306	**0.321**	**0.318**	**0.259**
HDP-HMM (VB)	0.312	0.291	0.309	0.305	0.244

Table 7.2: Comparison of predictive log likelihood (bits/spike) computed from 9 simulated data sets, measured in bits per spike improvement over a baseline of independent, homogeneous Poisson processes (the best result in each data set is marked in bold font).

Table 7.2 summarizes the predictive log likelihood comparison. For all five datasets, the HDP-HMM fit via Gibbs sampling with fixed κ_n performs best, though in general the increase over fitting the HDP-HMM when using HMC or EB for hyperparameter selection is small. By contrast, the improvement compared to fitting with VB inference or using a parametric HMM is quite significant.

Though computation cost is often a major factor with Bayesian inference, with the optimized PyHSMM package, the models can be fit to the synthetic data in under 10 minutes on an Apple MacBook Air. The runtime necessarily grows the number of neurons and the truncation limit on the number of latent states. As the model complexity grows, we must also run our MCMC algorithm for more iterations, which often motivates the use of variational inference algorithms instead. Given our optimized implementation and the performance improvements yielded by MCMC, we opted for a fully-Bayesian approach using MCMC with HMC for hyperparameter sampling in our subsequent experiments.

Figure 7.2 shows example traces from the MCMC combined with HMC algorithm for the HDP-HMM running on synthetic dataset 1. This is the same data from which Fig. 7.1 is generated. The first 5 Markov chain iterations have been omitted to highlight the variation in the latter samples (the first few iterations rapidly move away from the initial conditions). We see that the log likelihood of the data rapidly converges to nearly that of the true model (horizontal dotted line), and the number of states quickly converges to around $K = 35$. Note that the nuisance parameters α_0 and γ do not converge to the true values — this is due to the fact that the solution is insensitive to these parameters or the presence of local optimal. However, even the concentration parameters are different from the true values, they are still consistent with the inferred state transition matrix.

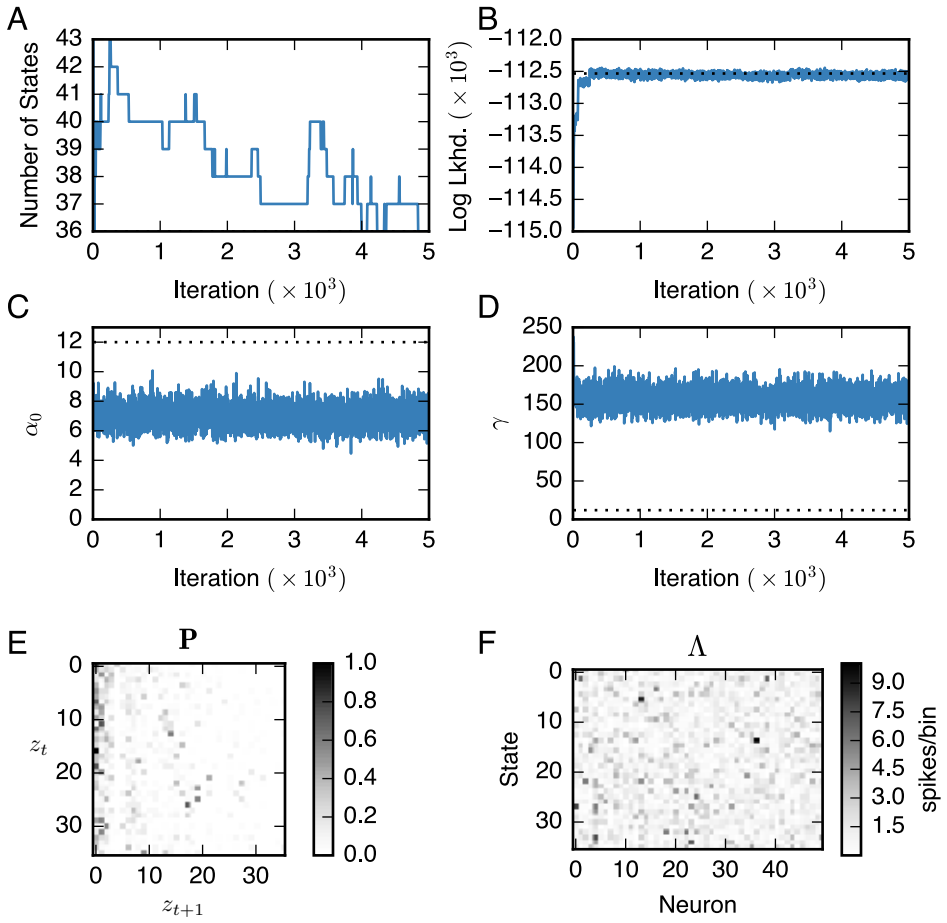

Figure 7.2: MCMC state trajectories for an HDP-HMM fit to the synthetic dataset shown in Fig. 7.1. True values are shown by the dotted black lines. The first five iterations of the Markov chain are omitted since they differ greatly from the final states. The chain quickly converges to nearly the correct number of states (A) and achieves close to the true log likelihood (B). (C, D) The chain trajectories of hyperparameters α_0 and γ. (E, F) Inferred state transition matrix and neuronal firing map drawn from the last iteration.

SENSITIVITY OF THE NUMBER OF LATENT STATES To test the sensitivity of the number of inferred states to changes in the data, we vary a number of parameters and plotted the number of inferred states in Fig. 7.3. In all cases, we use synthetic dataset 1, shown in Fig. 7.1, and HDP-HMMs fit via Gibbs sampling with fixed κ_n. First, we vary the number of observed neurons, N, and find that the number of inferred states was relatively stable around the true number of states ($K = 35$). By contrast, as we increase the observed recording length, T, the number of inferred states increases

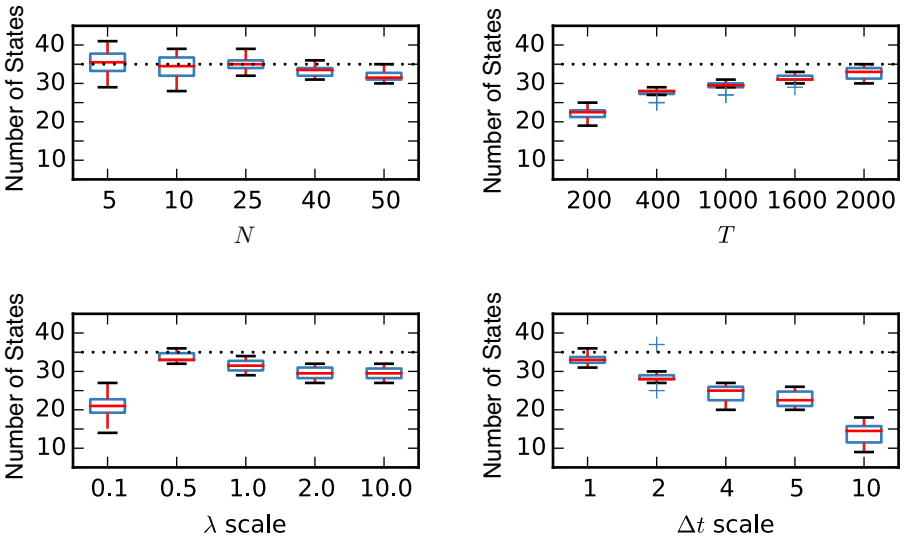

Figure 7.3: In a synthetic data experiment, we generated a spike train for a population of $N = 50$ neurons and $T_{\text{true}} = 2000$ time bins. Then we varied the number of observed neurons N, recording duration T, scale of the firing rate λ, temporal bin size Δt, and measured the number of inferred latent states. Horizontal dashed lines indicate the ground truth.

as well. This is because the true underlying data actually does visit more states as we simulate it for longer time. In general, we expect the number of inferred states to grow with the complexity of the data. Next, we vary the scale of the firing rate by multiplying the true model's firing rate by a factor of 0.1, 0.5, 1.0, 2.0, or 10.0, and sampling a new spike count. When the rates are very low, most bins do not contain any spikes, and hence it is not possible to resolve as many states. By contrast, when the rate is increased, the number of inferred states is slightly lower than the true number, which is likely the result of a slight mismatch with the prior on the firing rate scale (parameters μ and ν_0 in Section 7.2.2). Finally, we considerate the effect of time bin size by scaling up the bin sizes by factors of 2 through 10. For example, when scaling by a factor of 2, we add the spike counts in each pair of adjacent bins. This has a similar effect to decreasing the recording length by a factor of 2, and hence we see the number of inferred states decrease with bin size.

7.5 Hippocampal Place Cells

Next, we apply the proposed methods to experimental data of the rat hippocampus. This is the same dataset studied in previous chapters, but here we applied additional preprocessing. We bin

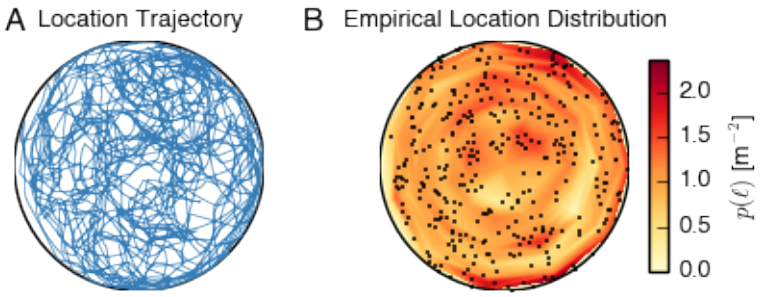

Figure 7.4: One rat's behavioral trajectory (left) and spatial occupancy (right) in the open field environment.

	Pred. log likelihood (bits/spike)	Decoding error (cm)
HMM ($K = 25$)	0.712	10.85 ± 6.43
HMM ($K = 45$)	0.706	10.71 ± 6.67
HMM ($K = 65$)	0.717	11.01 ± 6.93
HDP-HMM (Gibbs)	**0.722**	$\mathbf{9.56 \pm 5.31}$
HDP-HMM (HMC)	0.646	9.96 ± 6.05
HDP-HMM (EB)	0.579	10.81 ± 6.78
HDP-HMM (VB)	0.602	10.93 ± 6.24

Table 7.3: A comparison of HMMs, HDP-HMMs, and inference algorithms on the rat hippocampal data. Performance is measured in predictive log likelihood and mean decoding error on two minutes of held-out test data (the best result is marked in bold font).

the ensemble spike activity with a bin size of 250 ms and obtain the population vector $\boldsymbol{z}$ in time. To identify the period of rodent locomotion during spatial navigation, we use a velocity threshold ($>$ 10 cm/s) to select the RUN epochs and merge them together. The result is a recording that is 9.8 minutes in duration. One animal's RUN trajectory and spatial occupancy are shown in Fig. 7.4 (left and right panels, respectively). The empirical probability of a location, $p(\ell)$, is determined by dividing the arena into 220 bins of equal area (11 angular bins and 20 radial bins) and counting the fraction of time points in which the rat is in the corresponding bin.

In experimental data analysis, we focus on Bayesian nonparametric inference for HDP-HMM. For all methods, we increase the truncation level to a large value of $K_{\mathsf{max}} = 100$. To discover the model order of the variational solutions, we use the number of states visited by the most likely state sequence under the variational posterior. The MCMC algorithms yield samples of state sequences from which the model order can be directly counted.

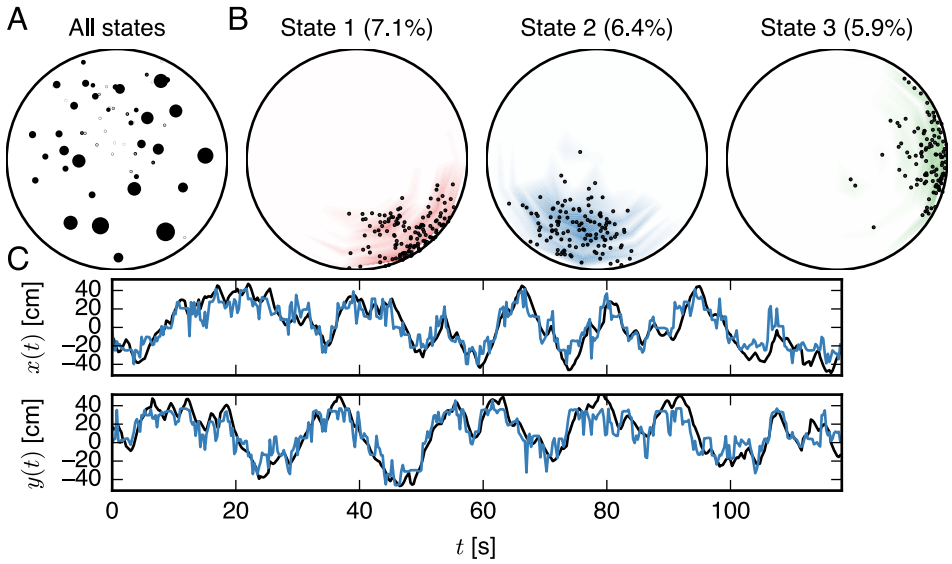

Figure 7.5: Estimation result from HDP-HMM (Gibbs) for the rat hippocampal ensemble spike data. (A) Estimated state space map, where the mean value of the spatial position for each latent state is shown by a black dot. The size of the dot is proportional to the occupancy of the state. (B) Probability distributions over location corresponding to the top three latent states, measured by state occupancy. The small black dots indicate the location of the animal while in that state, and are used to compute the empirical distribution over location indicated by colored shading. (C) The true and reconstructed trajectories in Cartesian coordinate. The true trajectory is shown in black and the reconstructed trajectory is shown in blue. For each time bin, we use the mean location of the latent states to determine an estimate of the animal's location.

We perform a quantitative comparison between HMMs, HDP-HMMs, inference algorithms, and hyperparameter setting algorithms, where performance is measured in terms of both decoding error and predictive log likelihood. For both metrics, we train the models on the first 7.8 minutes of data and test on the final two minutes of data for prediction. The results are summarized in Table 7.3. We find that the HDP-HMM fit by Gibbs sampling with fixed firing rate scale ($\kappa_n = 1$) again outperforms the competing models in both measures.

For the purpose of result assessment, we plot the state-space or state-location map (Fig. 7.5A), which shows the mean value of the spatial position that each state represented. The size of the black dot is proportional to the occupancy of the state. To compute an "empirical" distribution over locations for a given state, we first compute the posterior distribution over latent states with our inference algorithms. This gives us a set of probabilities $\Pr(z_t = k)$ for all time bins t and states k. Then we compute the average location for each state k by weighting the animal's location, (x_t, y_t)

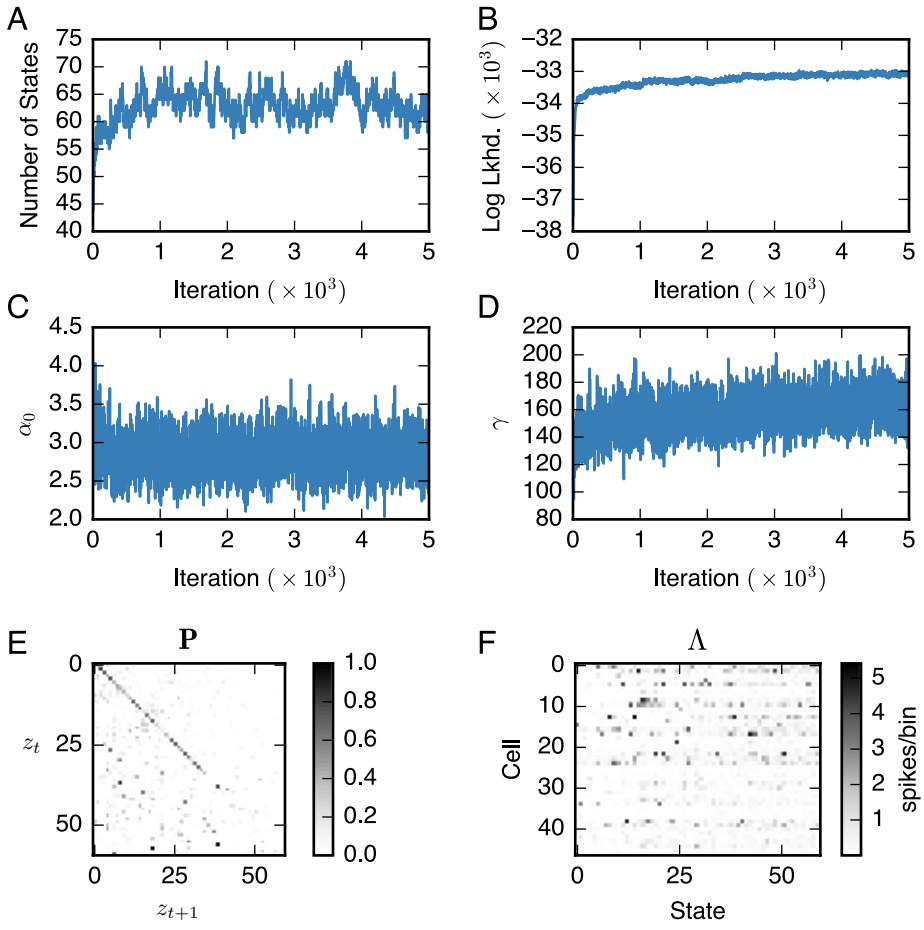

Figure 7.6: Estimation result from HDP-HMM (Gibbs) for the rat hippocampal ensemble spike data. (A) The total number of states (solid blue) slowly increases as states are allocated for a small number of time bins. The number of states converges after 2500 iterations. (B) The log likelihood of the training data grows consistently as highly specific states are added. (C, D) The concentration parameters, α_0 and γ also converge after 2500 iterations. (E, F) The inferred state transition matrix and firing rate samples drawn from the last iteration.

by the probability that the animal was in state k at time t. Summing over time yields a weighted set of locations, which we then bin into equal-area arcs and normalize to get an empirical distribution over locations for each state k.

The empirical location distribution for the top three states as measured by occupancy are shown in Fig. 7.5B). In Fig. 7.5C, we show the estimated animal's spatial trajectories in black, along with the reconstructed location in from the HDP-HMM with Gibbs sampling in blue. To reconstruct

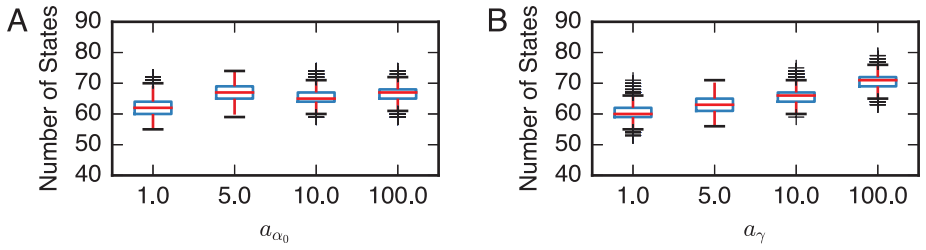

Figure 7.7: Measuring the effect of concentration hyperparameters on the number of inferred latent states. We find that the concentration hyperparameters of the gamma priors on the concentration parameters, α_0 and γ, have a minimal effect.

the position, we use the mean of each latent state's location distribution weighted by the marginal probability of that state under the HDP-HMM. That is,

$$\hat{x}_t = \sum_{k=1}^{K} \bar{x}_k \Pr(\boldsymbol{z}_t = k), \qquad \hat{y}_t = \sum_{k=1}^{K} \bar{y}_k \Pr(\boldsymbol{z}_t = k),$$

where $\bar{x}_k$ and $\bar{y}_k$ denote the average location of the rat while in inferred state k (corresponding to the black dots in Fig. 7.5A). Note that the animal's position is not used in model inference, only during result assessment. In the illustrated example (HDP-HMM with MCMC+HMC), the mean reconstruction error in Euclidean distance is 9.07 cm.

As the parameter sample traces in Fig. 7.6 show, the Markov chain converges in around 2500 iterations. After this point, the total number of states stabilizes to around 65. The concentration parameters α_0 and γ converge within a similar number of iterations. Finally, we show the transition matrix $\boldsymbol{P}$ and firing rate matrix $\boldsymbol{\Lambda}$ obtained from the final Markov chain sample.

We again evaluated the sensitivity of these model fits to the choice of hyperparameters. For the HDP-HMM fit via Gibbs sampling with fixed κ_n, the primary hyperparameters of interest are the concentration hyperparameters, a_{α_0} and a_γ in Eq. 7.3, where we have assumed $\alpha_0 \sim \mathrm{Gamma}(a_{\alpha_0}, 1)$ and $\gamma \sim \mathrm{Gamma}(a_\gamma, 1)$. Figure 7.7 shows the inferred number of states as we vary these two hyperparameters over orders of magnitude. We found that the number of inferred states is stable around 65, indicating the performance robustness to the choice of these hyperparameters.

Looking into the inferred states, we can reconstruct the "place fields" or "state fields" of hippocampal neurons. To do so, we combine the state-location maps (Fig. 7.5B) with the firing rate of

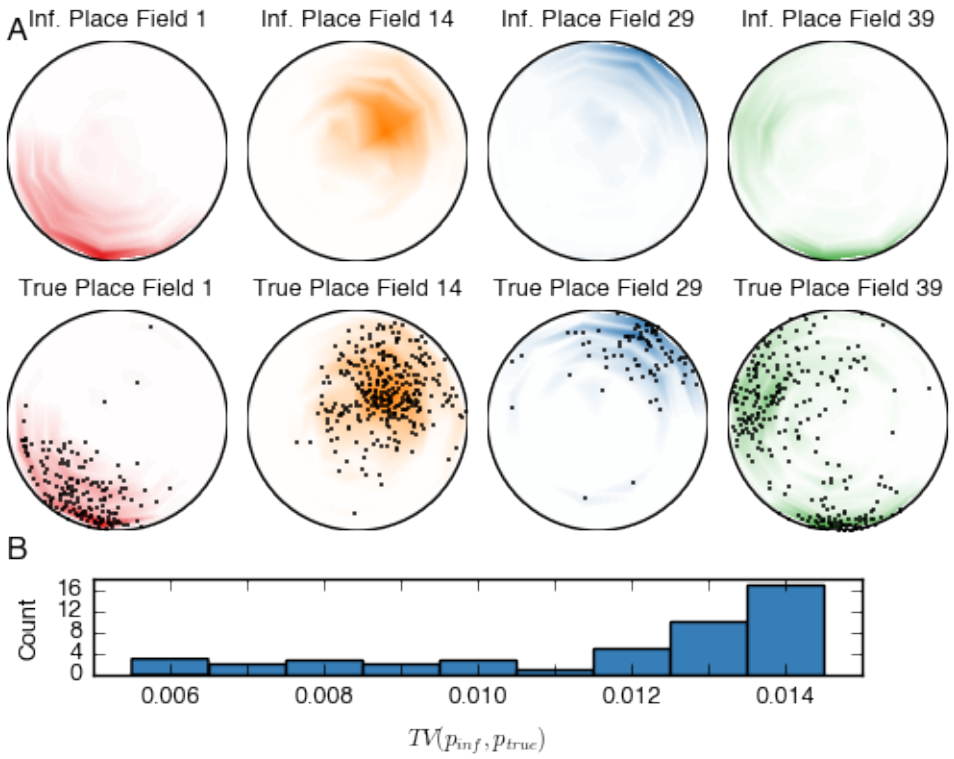

Figure 7.8: Comparison of inferred and true place fields for four randomly selected hippocampal neurons. The inferred place field (top row) for neuron n is a combination of location distributions for each state k weighted by the inferred firing rates $\lambda_{k,n}$, whereas the true place field (bottom row) for neuron n is a histogram of locations in which neuron n fires. The black dots show the rat's locations used for each histogram. The inferred place fields closely match the true place fields. With adequate spike data recording, we expect a higher latent state dimensionality to yield higher spatial resolution in the inferred place fields.

the individual neuron in those states (Fig. 7.6F) and weight by the marginal probability of the latent state. Together, these give rise to the inferred neuron's place field. Note that, again, the position data was only used in reconstruction but not in the inference procedure. Four pairs of inferred and true place fields are shown in Fig. 7.8. On the top row is the inferred place field; on the bottom is the true place field computed using the locations of the rat when neuron n fired shown by black dots.

7.6 Extensions

Hidden Semi-Markovian Models A striking feature of the inferred state transition matrix in Fig. 7.6E is that the first 40 states exhibit strong self-transitions. This is a common feature of time

series and has been addressed by a number of augmented Markovian models. In particular, hidden semi-Markovian models (HSMMs) explicitly model the duration of time spent in each state separately from the rest of the state transition matrix (Johnson and Willsky, 2013). Building this into the model allows the Dirichlet or HDP prior over state transition vectors to explain the rest of the transitions, which are often more similar. Alternatively, "sticky" HMMs and HDP-HMMs accomplish a similar effect (Fox et al., 2008).

Dependent Observation Models The HMMs in this chapter used conditionally independent Poisson observations. Given the latent state, each neuron fires independently of the others, and also independently of its previous spike counts. One way to extend these models is by introducing dependencies in the observation models. For example, we can combine the autoregressive models of previous chapters with the discrete latent states of an HMM with a model of the form,

$$p(\boldsymbol{s}_t \mid z_t) = \prod_{n=1}^{N} \text{Poisson}(s_{t,n} \mid \lambda_{t,n}),$$
$$\lambda_{t,n} = g\left(\psi_n^{(z_t)} + \boldsymbol{w}_n^{(z_t)} \boldsymbol{s}_{t-1}\right).$$

As in previous chapters, this can easily be extended to higher-order autoregressive models. Expectation-maximization algorithms for this type of model were developed by Escola et al. (2011). Alternatively, we can use the Pólya-gamma augmentation schemes of Chapter 5, and we have presented a preliminary versions of this approach in Johnson et al. (2015).

Input-Output HMMs Hidden Markov models are "open loop" systems: the next state depends only on the previous state. In practice, it is natural to expect that transitions are not only state-dependent but also a function of some external variables. For example, in the hippocampus where states correspond to actual locations, whether or not the rat transitions into a state may depend on instantaneous properties of that location. If more complex experimental setups there may be food or obstacles in the environment that affect where the rat goes next.

These types of external variables can be modeled with an input-output HMM (IOHMM) (Bengio and Frasconi, 1995). Suppose we have an external input, $\boldsymbol{u}_t \in \mathbb{R}^D$. We can model the transition

probability as,

$$\boldsymbol{\pi}^{(k)} \propto \exp\{\boldsymbol{\psi}^{(k)} + \boldsymbol{W}\boldsymbol{u}_t\},$$

that is, as a "soft-max" function of a baseline probability plus a weighted combination of input covariates. Performing inference in this type of model is not much more challenging than in the standard HMM. When sampling the latent states, we simply compute the instantaneous transition probabilities for each time step. In order to update the transition weights, $\boldsymbol{W}$, and the baseline probabilities, $\boldsymbol{\psi}^{(k)}$, we can either use HMC or our recently developed Pólya-gamma augmentation scheme for multinomial models (Linderman and Johnson, 2015).

7.7 Conclusion

This chapter explored the idea of dynamic latent states underlying neural activity. Specifically, we developed Bayesian nonparametric hidden Markov models (HDP-HMMs) and corresponding MCMC and variational inference algorithms. Since these models can be quite sensitive to hyperparameter settings, we performed a thorough assessment of inference results on both synthetic data and real recordings from rat hippocampal place cells. In the next chapter, we will build on these ideas, developing more sophisticated latent state space models with a mix of discrete and continuous latent states. As we will see, HMMs are only one in a hierarchy of state space models.

8

Switching Linear Dynamical Systems with Count Observations

The past two chapters have explored different notions of latent state: a dynamic network in Chapter 6 and a discrete latent state in Chapter 7. These states are a powerful addition to the autoregressive models of the earlier chapters. In this chapter, we consider one final extension — a continuous latent state that evolves over time. These continuous latent state space models are one of the most common methods in computational neuroscience (Smith and Brown, 2003; Paninski et al., 2010; Macke et al., 2011; Buesing et al., 2012a; Petreska et al., 2011; Cunningham and Yu, 2014).

The simplest form of continuous state space model assumes that the latent state obeys linear dynamics. Here, however, we will consider a more general case in which the dynamics are only *conditionally* linear given a dynamic discrete latent state (Petreska et al., 2011). This is known as a *switching* linear dynamical system (Murphy, 2012; Fox, 2009). By switching between different linear dynamical regimes, we obtain highly nonlinear patterns of dynamics. Moreover, this switching linear dynamical system will recover a number of common models as special cases.

The challenge, as should be expected by now, is in performing efficient inference in the face of discrete observations. The aforementioned existing methods have relied upon a Laplace approximation, which approximates the conditional distribution with a Gaussian. Given the tools developed in previous chapters, we can now develop asymptotically exact Gibbs sampling algorithms. In particular, the Pólya-gamma augmentations introduced in Chapter 5 will make it easy to develop effi-

cient algorithms that leverage many of the standard tools that exist for Gaussian observation models. Once we have augmented our observations with Pólya-gamma auxiliary variables, the observations are conditionally Gaussian distributed. Thus, all of our tools for efficient Bayesian inference in linear Gaussian models are at our disposal.

Finally, we will consider a problem that we have given little consideration thus far, namely, the problem of model comparison. We have tacitly assumed that predictive likelihoods provide a sufficient means of comparing two models. In practice, this has led to some difficulty, as we encountered with the network model comparison in Chapters 3 and 5. The root of the problem is that predictive likelihood comparisons only implicitly depend on model complexity. More complex models are more prone to overfitting, which should manifest itself in decreased predictive performance. However, there are more direct means of assessing the balance between model complexity and predictive capability. In theory, the marginal likelihood — the denominator in Bayes' rule — should provide a better estimate of the trade-off between how well a model fits the data and the size of the hypothesis class (MacKay, 1992; Kass and Raftery, 1995).

We will show how the conditionally conjugate models derived via Pólya-gamma augmentation enable principled marginal likelihood estimation with annealed importance sampling (AIS) (Neal, 2001). In order to make this practically feasible, however, we must dive into the inner workings of the Pólya-gamma distribution and develop a novel sampling algorithm capable of efficiently generating random variates in the "small shape" regime required by AIS.

8.1 A Hierarchy of Latent State Space Models

Consider a general class of models with a continuous latent state, $\boldsymbol{x}_t \in \mathbb{R}^D$, that obeys affine, but potentially nonstationary, dynamics at discrete time t,

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{b}_t,\ \boldsymbol{\Sigma}_t).$$

Let the initial state distribution have mean $\boldsymbol{\mu}_1$. Furthermore, assume a linear activation model $\boldsymbol{\psi}_t = \boldsymbol{C}\boldsymbol{x}_t$, where the mean spike count, $s_{t,n}$ is a nonlinear function of the activation, $\psi_{t,n}$, and neuron-specific parameters, ν_n. We refer to the collection of model parameters as,

$$\boldsymbol{\theta} = \left\{\{\boldsymbol{A}_t, \boldsymbol{b}_t, \boldsymbol{\Sigma}_t\}_{t=1}^T, \boldsymbol{\mu}_1, \boldsymbol{C}, \{\nu_n\}_{n=1}^N\right\}$$

Given these parameters, we can summarize this probabilistic model. In keeping with standard texts (e.g. Murphy, 2012, Chapter 18), we use "Matlab" notation to refer to a sequence of spike count vectors, $\boldsymbol{s}_{1:T}$, and a sequence of latent state vectors, $\boldsymbol{x}_{1:T}$. We have,

$$p(\boldsymbol{s}_{1:T}, \boldsymbol{x}_{1:T} \mid \boldsymbol{\theta}) = p(\boldsymbol{\theta})\, p(\boldsymbol{x}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{s}_{1:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{\theta})$$

where

$$\begin{aligned} p(\boldsymbol{x}_{1:T} \mid \boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \prod_{t=2}^{T} \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{b}_t, \boldsymbol{\Sigma}_t) \\ p(\boldsymbol{s}_{1:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{\theta}) &= \prod_{t=1}^{T} p(\boldsymbol{s}_t \mid \boldsymbol{C}\boldsymbol{x}_t, \{\nu_n\}) \\ &= \prod_{t=1}^{T} \prod_{n=1}^{N} p(s_{t,n} \mid \psi_{t,n}, \nu_n). \end{aligned} \tag{8.1}$$

Now consider the special case where there are only $K < T$ unique dynamics and covariance matrices, $\{\boldsymbol{A}_k, \boldsymbol{b}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, and that at any instant in time, the chosen dynamics are specified by the discrete latent variable $z_t \in \{1, \ldots, K\}$. Moreover, suppose this discrete latent variable follows a Markov model with initial state distribution $\boldsymbol{\pi}_0$ and transition probabilities $\{\boldsymbol{\pi}_k\}_{k=1}^K$, as in the last chapter. Then the dynamics for $\boldsymbol{z}_{1:T}$ and $\boldsymbol{x}_{1:T}$ are,

$$\begin{aligned} p(\boldsymbol{z}_{1:T} \mid \boldsymbol{\theta}) &= \text{Discrete}(z_1 \mid \boldsymbol{\pi}_0) \prod_{t=2}^{T} \text{Discrete}(z_t \mid \boldsymbol{\pi}_{z_{t-1}}). \\ p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{z_1}) \prod_{t=2}^{T} \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{A}_{z_t} \boldsymbol{x}_{t-1} + \boldsymbol{b}_{z_t}, \boldsymbol{\Sigma}_{z_t}), \end{aligned}$$

This is a *switching linear dynamical system* (SLDS) model (Murphy, 2012; Fox, 2009). At any point in time, the latent state obeys linear dynamics. The particular choice of dynamics switches between K discrete values according to a Markov model.

The SLDS contains a number of other models as special cases:

- When there is only one discrete latent state ($K = 1$), this reduces to a standard linear dynamical system (LDS).
- When there is one discrete latent state and no continuous dynamics ($\boldsymbol{A}_k \equiv 0$), this reduces

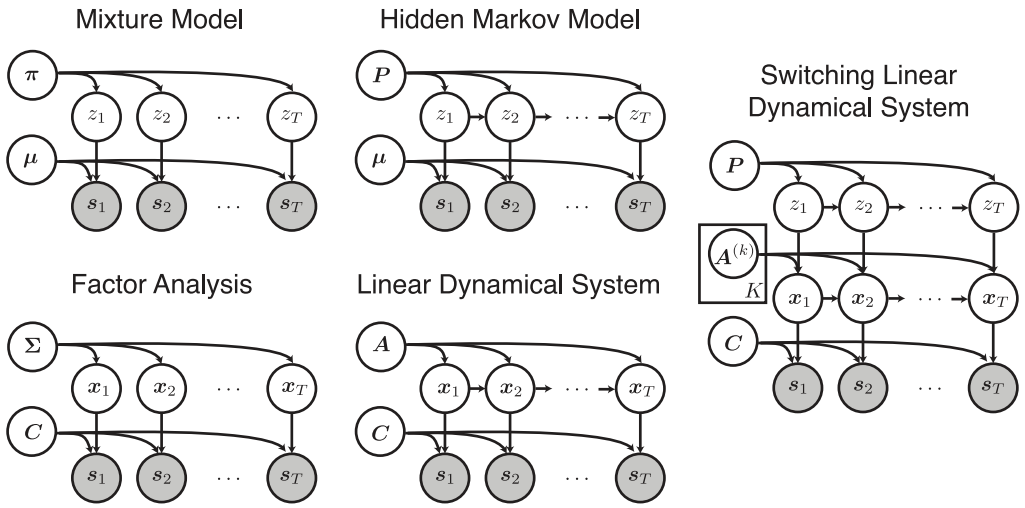

Figure 8.1: Special cases of the switching linear dynamical system. Adapted from Figure 2.2.

to factor analysis (FA).

- When (i) the state dimensionality is equal to the number of neurons ($D = N$); (ii) there are no continuous dynamics ($\boldsymbol{A}_k \equiv 0$); and (iii) the emission matrix is the identity ($\boldsymbol{C} \equiv \boldsymbol{I}$), the SLDS reduces to a hidden Markov model. At each point in time, the firing rate is determined solely by $\boldsymbol{b}_{z_t}$.

- When the the conditions of the HMM are met *and* the discrete transition matrix, $\boldsymbol{P}$, has identical rows ($\boldsymbol{\pi}_k \equiv \boldsymbol{\pi}_0$), the SLDS further reduces to a simple mixture model. At each point in time, the discrete latent state is drawn from $z_t \sim \mathrm{Discrete}(\boldsymbol{\pi}_0)$.

The graphical models corresponding to these special cases are shown in Figure 8.1, with the omission of some model parameters to conserve space. This figure is adapted from Figure 2.2. The only model that is not captured here is the autoregressive model since, here, all interaction between spike counts arises through the latent state. Next we show how a single, unified algorithm can support efficient inference in the SLDS and all its special cases.

8.2 Markov Chain Monte Carlo Inference

First we show how the continuous latent states, $\boldsymbol{x}_{1:T}$, can be updated with a block Gibbs sampler when the observations are conditionally Gaussian distributed. The key elements of the inference

algorithm will be conserved when we move to discrete count observations. Given the Gaussian inference algorithm, we will show how the Pólya-gamma augmentation explored in Chapter 5 enables efficient Bayesian inference in discrete models as well.

8.2.1 Block Gibbs Sampling Latent States with Gaussian Observations

Suppose the spike counts, $\boldsymbol{s}_t$ are conditionally distributed according to a Gaussian distribution. Moreover, assume the distribution has nonstationary precision, $\boldsymbol{\Omega}_t$, such that

$$p(\boldsymbol{s}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{s}_t \mid \boldsymbol{C}\boldsymbol{x}_t, \boldsymbol{\Omega}_t^{-1}). \tag{8.2}$$

In this case, the conditional density over continuous latent states, $p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta})$, is jointly Gaussian as well. We perform a block Gibbs update for the entire latent state sequence, $\boldsymbol{x}_{1:T}$, using a forward filtering-backward sampling algorithm, just as we did for the HMM in Section 7.2.1.

The marginal "filtered" distribution given observations up to time t is a Gaussian, which we will denote by,

$$p(\boldsymbol{x}_t \mid \boldsymbol{s}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{m}_t, \boldsymbol{V}_t),$$

where $\boldsymbol{m}_t$ and $\boldsymbol{V}_t$ are the filtered mean and covariance, respectively. Kalman filtering is an iterative for computing the filtered means and variances of a Gaussian linear dynamical system, and it is analogous to the HMM filtering algorithms of the previous chapter. Here, we follow the presentation of Murphy (2012, Chapter 18). Kalman filtering consists of iterating forward in time from $t = 1$ to $t = T$. Assume that at iteration t we have already computed $\boldsymbol{m}_{t-1}$ and $\boldsymbol{V}_{t-1}$. As with the HMM, given the Markovian structure of the probabilistic model, the conditional distribution of $\boldsymbol{x}_t$ factors into,

$$p(\boldsymbol{x}_t \mid \boldsymbol{s}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{\theta}) \propto \underbrace{p(\boldsymbol{s}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})}_{\text{condition}} \underbrace{p(\boldsymbol{x}_t \mid \boldsymbol{s}_{1:t-1}, \boldsymbol{z}_{1:t}, \boldsymbol{\theta})}_{\text{predict}}.$$

We will show that both of these factors are Gaussian distributions, and hence their product is as well.

The first step is to *predict* $\boldsymbol{x}_t$ given observations $\boldsymbol{s}_{1:t-1}$. To do so, we marginalize over the previ-

ous latent state, $\boldsymbol{x}_{t-1}$,

$$\begin{aligned} p(\boldsymbol{x}_t \mid \boldsymbol{s}_{1:t-1}, \boldsymbol{z}_{1:t}, \boldsymbol{\theta}) &\propto \int p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, z_t, \boldsymbol{\theta})\, p(\boldsymbol{x}_{t-1} \mid \boldsymbol{s}_{1:t-1}, \boldsymbol{z}_{1:t-1}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{x}_{t-1} \\ &= \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{m}_{t|t-1}, \boldsymbol{V}_{t|t-1}), \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{m}_{t|t-1} &\triangleq \boldsymbol{A}_t \boldsymbol{m}_{t-1} + \boldsymbol{b}_t, \\ \boldsymbol{V}_{t|t-1} &\triangleq \boldsymbol{A}_t \boldsymbol{V}_{t-1} \boldsymbol{A}_t^\mathsf{T} + \boldsymbol{\Sigma}_t. \end{aligned}$$

Then, we *condition* on the current observations, $\boldsymbol{s}_t$, to get the parameters of the filtered distribution,

$$\begin{aligned} \boldsymbol{m}_t &= \boldsymbol{m}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{s}_t - \boldsymbol{C}\boldsymbol{m}_{t|t-1}), \\ \boldsymbol{V}_t &= (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C})\boldsymbol{V}_{t|t-1}, \end{aligned} \tag{8.3}$$

where $\boldsymbol{K}_t$ is the "Kalman gain" matrix,

$$\boldsymbol{K}_t \triangleq \boldsymbol{V}_{t|t-1}\boldsymbol{C}^\mathsf{T}\left[\boldsymbol{C}\boldsymbol{V}_{t|t-1}\boldsymbol{C}^\mathsf{T} + \boldsymbol{\Omega}_t^{-1}\right]^{-1}.$$

Once we have computed the filtered means and covariances for all time bins, we can sample from the joint distribution over $\boldsymbol{x}_{1:T}$ by applying the chain rule,

$$\begin{aligned} p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) &= p(\boldsymbol{x}_T \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \prod_t p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1:T}, \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \\ &\propto p(\boldsymbol{x}_T \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \prod_t p(\boldsymbol{x}_t \mid \boldsymbol{s}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{\theta})\, p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, z_{t+1}, \boldsymbol{\theta}). \end{aligned}$$

Thus, we can sample in reverse order, starting with $\boldsymbol{x}_T$ and ending with $\boldsymbol{x}_1$. The conditional distribution of $\boldsymbol{x}_t$

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1:T}, \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \propto \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{m}_t, \boldsymbol{V}_t)\, \mathcal{N}(\boldsymbol{x}_{t+1} \mid \boldsymbol{A}_{t+1}\boldsymbol{x}_t + \boldsymbol{b}_{t+1}, \boldsymbol{\Sigma}_{t+1}), \tag{8.4}$$

which is yet another Gaussian distribution. Now we can write the complete algorithm for block Gibbs sampling the continuous latent states, $\boldsymbol{x}_{1:T}$.

Require: $\boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}$
  for $t = 1, \ldots, T$ **do**
    Compute $\boldsymbol{m}_t$ and $\boldsymbol{V}_t$ ▷ Eq. 8.3
  end for
  for $t = T, \ldots, 1$ **do**
    Sample $\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}, \boldsymbol{m}_t, \boldsymbol{V}_t, \boldsymbol{\theta}$ ▷ Eq. 8.4
  end for

Algorithm 8.1: Forward filtering-backward sampling (FFBS) algorithm for the Gaussian linear dynamical system. Note the similarity to the FFBS algorithm for HMMs in Alg. 7.1.

8.2.2 Pólya-gamma Augmentation for Discrete Observations

The conditional distribution of the latent states is only Gaussian if the observations are as well. Fortunately, the observations become conditionally Gaussian after augmenting the data with Pólya-gamma auxiliary variables. Recall from Chapter 5 that the Pólya-gamma augmentation is an auxiliary variable scheme that applies to models with logistic link functions (Polson et al., 2013). Specifically, this augmentation can be used to develop Gibbs for models with likelihoods of the form,

$$\begin{aligned} p(s \mid \psi, \nu) &= c(s, \nu)\, \sigma(\psi)^{a(s,\nu)}\, (1 - \sigma(\psi))^{d(s,\nu)} \\ &= c(s, \nu) \frac{(e^{\psi})^{a(s,\nu)}}{(1 + e^{\psi})^{b(s,\nu)}}. \end{aligned}$$

These are called *logistic likelihoods* because the latent variables are transformed by a logistic function, $\sigma(\psi) = e^{\psi}/(1 + e^{\psi})$. Bernoulli, binomial, negative binomial, and multinomial likelihoods can all be put in this form. For example, in the Bernoulli case,

$$\mathrm{Bern}(s \mid \psi) = \sigma(\psi)^s (1 - \sigma(\psi))^{1-s} = \frac{(e^{\psi})^s}{(1 + e^{\psi})}.$$

Thus, $a(s, \nu) = s$, $b(s, \nu) \equiv 1$, and $c(s, \nu) \equiv 1$. We refer the reader back to Table 5.1 for the formulation of other count distributions.

The augmentation is based on an integral identity derived from the Laplace transform of the Pólya-gamma density. If $p_{\mathrm{PG}}(\omega \mid b, 0)$ is the density of the Pólya-gamma distribution, $\mathrm{PG}(b, 0)$, then,

$$\frac{(e^{\psi})^a}{(1 + e^{\psi})^b} = 2^{-b} e^{\kappa \psi} \int_0^{\infty} e^{-\omega \psi^2 / 2}\, p_{\mathrm{PG}}(\omega \mid b, 0)\, \mathrm{d}\omega, \tag{8.5}$$

where $\kappa = a - b/2$. The integral on the right-hand side is the Laplace transform of the Pólya-gamma density evaluated at $\psi^2/2$, and the left-hand side is the same form found in discrete distributions with logistic link functions. Importantly, viewed as a function of ψ for fixed ω, the right-hand side is an unnormalized Gaussian density. Thus, the identity in (8.5) transforms a logistic likelihood to a Gaussian likelihood conditioned on an auxiliary variable, ω.

Now, let us return to the likelihood of (8.1), where $\psi_{t,n} = [\boldsymbol{C}\boldsymbol{x}_t]_n = \boldsymbol{c}_n^\mathsf{T}\boldsymbol{x}_t$ is the activation of neuron n at time t. As a function of $\boldsymbol{x}_t$, the likelihood is proportional to,

$$\begin{aligned} p(\boldsymbol{s}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta}) &\propto \prod_{n=1}^N \frac{(e^{\psi_{t,n}})^{a(s_{t,n},\nu_n)}}{(1+e^{\psi_{t,n}})^{b(s_{t,n},\nu_n)}} \\ &\propto \prod_{n=1}^N e^{\kappa(s_{t,n},\nu_n)\psi_{t,n}} \int_0^\infty e^{-\omega_{t,n}\psi_{t,n}^2/2}\, p_{\mathrm{PG}}(\omega_{t,n} \mid b(s_{t,n},\nu_n), 0)\, \mathrm{d}\omega_{t,n}. \end{aligned}$$

By introducing $\omega_{t,n}$ as auxiliary variables, the likelihood of $\boldsymbol{x}_t$ is proportional to a multivariate Gaussian distribution,

$$\begin{aligned} p(\boldsymbol{s}_t \mid \boldsymbol{x}_t, \boldsymbol{\omega}_t, \{\nu_n\}) &\propto \prod_{n=1}^N \mathcal{N}(\boldsymbol{c}_n^\mathsf{T}\boldsymbol{x}_t \mid \omega_{t,n}^{-1}\kappa(s_{t,n},\nu_n),\, \omega_{t,n}^{-1}) \\ &\propto \mathcal{N}(\widehat{\boldsymbol{s}}_t \mid \boldsymbol{C}\boldsymbol{x}_t,\, \boldsymbol{\Omega}_t^{-1}), \end{aligned} \tag{8.6}$$

where

$$\begin{aligned} \boldsymbol{\kappa}_t &= [\kappa(s_{t,1},\nu_1), \ldots, \kappa(s_{t,N},\nu_N)]^\mathsf{T} \\ \widehat{\boldsymbol{s}}_t &= \boldsymbol{\Omega}_t^{-1}\boldsymbol{\kappa}_t \\ \boldsymbol{\Omega}_t &= \mathrm{diag}\left([\omega_{t,1}, \ldots, \omega_{t,N}]\right). \end{aligned}$$

Note the similarity between the augmented likelihood of (8.6) and the Gaussian likelihood of (8.2). The only difference is that, here, the precision is given by the auxiliary variables, and the "effective" observations, $\widehat{s}_{t,n}$, are a function of $s_{t,n}$, $\omega_{t,n}$, and ν_n. Thus, given a set of Pólya-gamma auxiliary variables, the block Gibbs updates in Algorithm 8.1 will apply equally well to the setting with discrete count observations.

Moreover, by the exponential tilting property of the Pólya-gamma distribution, the conditional

distribution of $\omega_{t,n}$ is proportional to a Pólya-gamma distribution:

$$\begin{aligned} p(\omega_{t,n} \mid \psi_{t,n}, s_{t,n}, \nu_n) &\propto e^{-\omega_{t,n}\psi_{t,n}^2/2}\, p_{\mathrm{PG}}(\omega_{t,n} \mid b(s_{t,n}, \nu_n),\, 0) \\ &\propto p_{\mathrm{PG}}(\omega_{t,n} \mid b(s_{t,n}, \nu_n),\, \psi_{t,n}). \end{aligned} \tag{8.7}$$

These auxiliary variables are conditionally independent of each other, and hence amenable to block parallel Gibbs sampling. Efficient Pólya-gamma sampling algorithms have been developed for the regimes typically encountered in Bernoulli, binomial, and negative binomial models (Windle et al., 2014).

The proposed algorithm for sampling the latent variables and parameters of an SLDS is summarized in Algorithm 8.2.

Require: $\boldsymbol{s}_{1:T}$ and $\boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T}$, and $\boldsymbol{\theta}$ from previous iteration
  Sample $\boldsymbol{\theta} \mid \boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T}, \boldsymbol{s}_{1:T}$
  Sample $\boldsymbol{z}_{1:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{\theta}$ ▷ Algorithm 7.1
  for $t = 1, \ldots, T$ **do** ▷ In parallel
    for $n = 1, \ldots, N$ **do** ▷ In parallel
      Sample $\omega_{t,n} \mid s_{t,n}, \boldsymbol{x}_t, \boldsymbol{\theta}$ ▷ Eq. 8.7
    end for
  end for
  Compute $\boldsymbol{\Omega}_{1:T}$ and $\widehat{\boldsymbol{s}}_{1:T}$ ▷ Eq. 8.6
  Sample $\boldsymbol{x}_{1:T} \mid \widehat{\boldsymbol{s}}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\Omega}_{1:T}, \boldsymbol{\theta}$ ▷ Algorithm 8.1

Algorithm 8.2: Single iteration of Gibbs sampler for an switching LDS with discrete count observations.

8.2.3 Missing Data

Sometimes we only have partial observations. For example, in some cases we have multiple recordings from the same circuit, but each recording only provides access to a subset of the population of neurons (Turaga et al., 2013). In other cases, we simply hold out some of the data for predictive likelihood comparisons. With Gaussian observations, we can implement this by replacing the missing data point, $s_{t,n}$, with a zero mean, zero precision observation. In the discrete count model, this can be implemented by setting the auxiliary variable, $\omega_{t,n}$, and the effective observation, $\widehat{s}_{t,n}$, to zero. Recall that the Pólya-gamma auxiliary variables specify the precision of the effective observations. By setting this to zero, we effectively remove this data point.

8.3 Alternative Approaches

Most alternative approaches to performing Bayesian inference in latent state space models with discrete observations have relied on a Laplace approximation (Tierney and Kadane, 1986) to the conditional distribution, $p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta})$ (Smith and Brown, 2003; Paninski et al., 2010; Macke et al., 2011). * Given a Gaussian approximation, the model parameters, $\boldsymbol{\theta}$, can be optimized such that they maximize the *expected* joint log probability under the approximate Gaussian distribution on $\boldsymbol{x}_{1:T}$. This constitutes an approximate expectation-maximization (EM) algorithm (Dempster et al., 1977).

For completeness, we describe the fundamentals of this approach, largely following the presentation of Macke et al. (2011). Consider a generative model in which $s_{t,n} \sim \text{Poisson}(\exp\{\boldsymbol{c}_n^\mathsf{T} \boldsymbol{x}_t\})$. The conditional log probability of $\boldsymbol{x}_{1:T}$ is given by,

$$\begin{aligned}\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \simeq &\sum_{t=1}^{T}\sum_{n=1}^{N} s_{t,n}(\boldsymbol{c}_n^\mathsf{T}\boldsymbol{x}_t) - \exp\{\boldsymbol{c}_n^\mathsf{T}\boldsymbol{x}_t\} + \\ &- \frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^\mathsf{T}\boldsymbol{\Sigma}_{z_1}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) + \\ &- \frac{1}{2}\sum_{t=2}^{T}(\boldsymbol{x}_t - \boldsymbol{A}_{z_t}\boldsymbol{x}_{t-1} - \boldsymbol{b}_{z_t})^\mathsf{T}\boldsymbol{\Sigma}_{z_t}^{-1}(\boldsymbol{x}_t - \boldsymbol{A}_{z_t}\boldsymbol{x}_{t-1} - \boldsymbol{b}_{z_t}),\end{aligned}$$

where $\simeq$ denotes equality up to an additive constant. This log probability is concave and can be efficiently maximized to obtain the mean of the Laplace approximation,

$$\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{x}_{1:T}} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}).$$

Once the mean has been found, the optimal covariance is given by the inverse Hessian of the log posterior evaluated at $\boldsymbol{\mu}^*$,

$$\boldsymbol{\Sigma}^* = -\left[\nabla^2_{\boldsymbol{x}_{1:T}} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta})\Big|_{\boldsymbol{x}_{1:T}=\boldsymbol{\mu}^*}\right]^{-1}.$$

By exploiting the chain structure of the graphical model, this inverse Hessian can be computed in time linear in T using essentially the same forward-backward approaches used during sampling.

*While the Laplace approximation is most common, see Buesing et al. (2012b) and Pfau et al. (2013) for some interesting new directions.

The mean and covariance parameterize a Gaussian approximation,

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta}) \approx q(\boldsymbol{x}_{1:T}) = \mathcal{N}(\boldsymbol{x}_{1:T} \mid \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*).$$

Given this approximation, the parameters are updated by maximizing the expected log probability,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_q \left[\log p(\boldsymbol{s}_{1:T}, \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta})\right].$$

For example, consider this expectation as a function of the emission matrix, $\boldsymbol{C}$,

$$\begin{aligned}
\mathbb{E}_q \left[\log p(\boldsymbol{s}_{1:T}, \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{\theta})\right] &\simeq \sum_{t=1}^{T} \sum_{n=1}^{N} s_{t,n} (\boldsymbol{c}_n^\mathsf{T} \mathbb{E}_q[\boldsymbol{x}_t]) - \mathbb{E}_q \left[\exp\{\boldsymbol{c}_n^\mathsf{T} \boldsymbol{x}_t\}\right], \\
&= \sum_{t=1}^{T} \sum_{n=1}^{N} s_{t,n} (\boldsymbol{c}_n^\mathsf{T} \boldsymbol{\mu}_t^*) - \exp\left\{\boldsymbol{c}_n^\mathsf{T} \boldsymbol{\mu}_t^* + \frac{1}{2} \boldsymbol{c}_n^\mathsf{T} \boldsymbol{\Sigma}_{tt}^* \boldsymbol{c}_n\right\}.
\end{aligned}$$

The last line follows from the moment generating function of the multivariate Gaussian distribution. This objective function is concave in $\boldsymbol{c}_n$. Note that closed form, concave expectations arise from the particular choice of exponential link function. Other models may require Monte Carlo estimates of the expectation inside the optimization. As more and more approximation is required, the performance of these methods tends to suffer.

Finally, we must handle the discrete latent states, $\boldsymbol{z}_{1:T}$. The simplest approach would be to alternate between updating the discrete and continuous latent states, as in the MCMC algorithm presented above. Alternatively, Petreska et al. (2011) have suggested a joint update for both $\boldsymbol{x}_{1:T}$ and $\boldsymbol{z}_{1:T}$ based on an approximate filtering technique.

From our perspective, the principal advantages of the Pólya-gamma augmentation are: (i) it allows for simple block Gibbs updates that leverage off-the-shelf code for Gaussian models; (ii) it provides an asymptotically unbiased MCMC algorithm; (iii) the stochasticity of the MCMC transitions allows the sampling algorithm to escape local modes, to which expectation-maximization algorithms are prone (Bishop, 2006); and (iv) once we have an MCMC algorithm, a number of natural extensions are clear, like the marginal likelihood estimation methods we discuss next.

8.4 Model Comparison via Marginal Likelihood Estimation

The marginal likelihood is the probability of the data, $\boldsymbol{s}$, having integrated out the latent variables, $\boldsymbol{z}$ and $\boldsymbol{x}$, and the parameters, $\boldsymbol{\theta}$,

$$p(\boldsymbol{s}) = \int p(\boldsymbol{s} \mid \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta})\, p(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{\theta}.$$

By integrating over the latent variables and parameters, the marginal likelihood captures a tradeoff between a model's complexity and its ability to explain the data. As such, it is a natural criterion for model comparison. In some cases, like linear Gaussian models with Gaussian observations, the marginal likelihood can be computed exactly. In these cases, marginal likelihood is often the gold-standard for model selection (Kass and Raftery, 1995; Grosse et al., 2015).

Unfortunately, seemingly small changes to the model can render the integration over parameters and latent variables intractable. For example, in linear Gaussian models with discrete observations, the marginal likelihood is no longer tractable. Instead, we must resort to approximate methods like annealed importance sampling (AIS) (Neal, 2001). AIS is based on sampling from a sequence of intermediate distributions that *anneal* between a tractable distribution and the intractable posterior. While AIS has proven highly effective for a variety of models (Grosse et al., 2015), the accuracy of the method hinges upon the efficiency of the Markov transition operators that target the intermediate distributions. Unfortunately, while the posterior distribution may admit efficient MCMC algorithms, the intermediate distributions may not. We show how the Pólya-gamma augmentation strategies above can be extended to perform efficient annealed importance sampling in the class of switching linear dynamical systems models with count observations.

8.4.1 Annealed Importance Sampling

AIS starts with a sample from a tractable distribution with a computable normalization constant. The prior distribution often suffices. Given this initial sample, a sequence of Markov transition operators is applied. The stationary distributions of these transition operators interpolate between the tractable initial distribution and the intractable posterior. The posterior density is proportional to the joint density, and the normalizing constant is the marginal likelihood of interest. Formally, the annealing path is a sequence of distributions, $q_1(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})$ to $q_M(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{s})$, where

$$q_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) = \frac{f_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})}{\mathcal{Z}_m}, \qquad f_M(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) = p(\theta, \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{s}), \qquad \mathcal{Z}_M = p(\boldsymbol{s}).$$

Let $q_1(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})$ be the normalized prior distribution such that $f_1(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})$ and $\mathcal{Z}_1 = 1$. Then, let $f_m(\boldsymbol{z}, \theta)$ be a geometric average of the prior and the joint:

$$\begin{aligned} f_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) &= \Big[p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\Big]^{1-\beta_m} \Big[p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{s})\Big]^{\beta_m} \\ &= p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\, p(\boldsymbol{s} \mid \boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})^{\beta_m}, \end{aligned}$$

with β_m monotonically increasing from $\beta_1 = 0$ to $\beta_M = 1$. As we anneal between $\beta = 0$ and $\beta = 1$, the intermediate distributions interpolate between the prior and the posterior. This geometric path is most common, but any path that starts with a tractable distribution and ends with the posterior will suffice (e.g. Grosse et al., 2013).

In addition to a annealing path, we also need a sequence of MCMC transition operators that leave the intermediate distributions q_m invariant,

$$\mathcal{T}_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x} \to \boldsymbol{\theta}', \boldsymbol{z}', \boldsymbol{x}').$$

Starting with a sample from the prior and applying these transition operators for $m = 1, \ldots, M$ yields a sample that is closer in distribution to the posterior. AIS uses this procedure as a proposal distribution for importance sampling. The importance weights are given by a product of ratios between f_m and f_{m-1}. Since the target density is the unnormalized posterior density, the importance weights will be unbiased estimates of the normalization constant, namely the marginal likelihood, $\mathcal{Z}_M = p(\boldsymbol{s})$. The annealed importance sampling algorithm is summarized in Algorithm 8.3.

for $\ell = 1$ to L **do**
  $w^{(\ell)} \leftarrow \mathcal{Z}_1$
  Sample $\boldsymbol{\theta}^{(1)}, \boldsymbol{z}^{(1)}, \boldsymbol{x}^{(1)} \sim q_1(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})$
  for $m = 2$ to M **do**
    $w^{(\ell)} \leftarrow w^{(\ell)} \times \frac{f_m(\boldsymbol{\theta}^{(m-1)}, \boldsymbol{z}^{(m-1)}, \boldsymbol{x}^{(m-1)})}{f_{m-1}(\boldsymbol{\theta}^{(m-1)}, \boldsymbol{z}^{(m-1)}, \boldsymbol{x}^{(m-1)})}$
    Sample $\boldsymbol{\theta}^{(m)}, \boldsymbol{z}^{(m)}, \boldsymbol{x}^{(m)} \sim \mathcal{T}_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x} \leftarrow \boldsymbol{\theta}^{(m-1)},\ \boldsymbol{z}^{(m-1)},\ \boldsymbol{x}^{(m-1)})$
  end for
end for
return $\widehat{\mathcal{Z}}_M = \frac{1}{L} \sum_{\ell=1}^{L} w^{(\ell)}$

Algorithm 8.3: Annealed Importance Sampling (AIS). Adapted from (Grosse et al., 2015).

How can we reduce the variance of this estimator? First, we can increase the number of interme-

diate distributions; second, we can design rapidly mixing transition operators, $\mathcal{T}_m$. In this section, we develop transition operators that are both computationally efficient, allowing us to run more transitions in a fixed amount of time, and more effective, in that they reach the equilibrium distribution more quickly.

With a geometric annealing path, the intermediate distributions of the switching LDS are given by,

$$f_m(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x}) \prod_{t=1}^{T} \prod_{n=1}^{N} c(s_{t,n}, \nu_n)^{\beta_m} \frac{(e^{\psi_{t,n}})^{a(s_{t,n},\nu_n)\cdot\beta_m}}{(1+e^{\psi_{t,n}})^{b(s_{t,n},\nu_n)\cdot\beta_m}}. \tag{8.8}$$

where, again, ν_n is a parameter in $\boldsymbol{\theta}$, and $\psi_{t,n}$ is a function of $\boldsymbol{x}$ and $\boldsymbol{\theta}$. Raising the likelihood to the power β_m does change its functional form; it only changes the power in the exponent. Most importantly, it is still amenable to Pólya-gamma augmentation! Thus, the Gibbs sweep defined in Algorithm 8.2 can be used as a transition operator, $\mathcal{T}_m$. The only differences in targeting f_m are that,

$$\kappa(s_{t,n}, \nu_n) = \left(a(s_{t,n}, \nu) - \frac{1}{2} b(s_{t,n}, \nu_n)\right) \cdot \beta_m,$$

and

$$p(\omega_{t,n} \mid \psi_{t,n}, s_{t,n}, \nu_n) \propto p_{\mathrm{PG}}(\omega_{t,n} \mid \underbrace{b(s_{t,n}, \nu_n) \cdot \beta_m}_{\text{often } <1}, \psi_{t,n}).$$

This provides some intuition into how AIS works. When β_m approaches zero, the intermediate distribution reduces to the prior. This is equivalent to setting κ and ω to zero. As $b \to 0$, the density Pólya-gamma density, $\mathrm{PG}(\omega \mid b, \psi)$, approaches a delta function at zero.

Note, however, that in order to implement $\mathcal{T}_m$ efficiently, we must be able to sample from the Pólya-gamma conditional distribution in the regime where $b(s_{t,n}, \nu_n) \cdot \beta_m < 1$. For Bernoulli observations, $b(s_{t,n}, \nu_n) \equiv 1$, so we will be in this regime for all $\beta_m \in [0, 1)$. While efficient samplers exist for Pólya-gamma distributed variables when $b(s_{t,n}, \nu_n) \cdot \beta_m \geq 1$ (Windle et al., 2014), the default method for this "small shape" regime is to approximate a Pólya-gamma sample with a truncated sum of gamma random variates (Polson et al., 2013). As the number of random variates in the sum approaches infinity, the approximate sample converges to a true draw from the Pólya-gamma distribution. To get a reasonably accurate draw, we typically need to sample around 200

gamma random variates per Pólya-gamma sample. With TN auxiliary variables, this quickly becomes prohibitively expensive. Next, we develop a novel sampling algorithm that makes these conditional updates very efficient, and renders AIS with Pólya-gamma augmented transitions highly effective.

8.5 A Novel Sampling Algorithm for the Pólya-gamma Distribution

The Pólya-gamma distribution, $\mathrm{PG}(b, \psi)$, is closely related to the Jacobi distribution, $J^*(b, \psi)$, surveyed by Biane et al. (2001) and elaborated upon in Windle et al. (2014). Specifically,

$$Y \sim J^*(b, \tfrac{\psi}{2}) \implies \frac{1}{4} Y \sim \mathrm{PG}(b, \psi).$$

Thus, to develop a sampler for the Pólya-gamma distribution, it is sufficient to be able to sample the Jacobi distribution.

As derived by Windle et al. (2014), the density of $J^*(b, \psi)$ can be written as an infinite alternating sum,

$$p_{J^*}(\omega \mid b, \psi) = \cosh^b(\psi) e^{-\omega\psi^2/2} \frac{2^b}{\Gamma(b)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(n+b)}{\Gamma(n+1)} \frac{(2n+b)}{\sqrt{2\pi\omega^3}} \exp\left\{ -\frac{(2n+b)^2}{2\omega} \right\}. \quad (8.9)$$

Windle et al. (2014) developed a number of methods for sampling this distribution. Most rely on finding tractable upper bounds on the density that can serve as a proposal distribution. Given a sample from the proposal, it is possible to accept or reject using the *alternating series method* (Devroye, 1986). We will go into more detail on this shortly.

We take the same basic approach, but we present a novel means of finding an upper bound on the Jacobi density. Massaging terms in (8.9), we can factor it into the product of three terms:

$$p_{J^*}(\omega \mid b, \psi) = \alpha^{-1}(b, \psi)\, p_{\mathrm{IG}}\left(\omega \,\middle|\, \frac{b}{|\psi|}, b^2 \right) \Phi(\omega \mid b). \quad (8.10)$$

The first term, $\alpha^{-1}(b, \psi)$, is a scaling constant greater than one,

$$\alpha^{-1}(b, \psi) = 2^b \cosh^b(\psi)\, e^{-b|\psi|} = \left(1 + e^{-2|\psi|}\right)^b \geq 1.$$

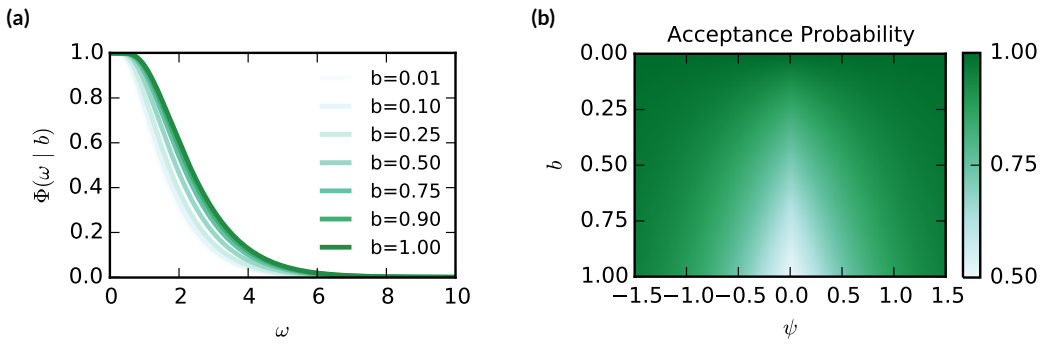

Figure 8.2: (a) Plot of $\Phi(\omega \mid b)$, the conditional acceptance probability for a proposed value of ω, for a range of $b \in (0, 1]$. In all cases, this function is monotonically decreasing from 1 to 0 as a function of ω, and thus defines a cumulative distribution function. **(b)** Acceptance probability, $\alpha(b, \psi)$, as a function of b and ψ.

The second is an inverse Gaussian density,

$$p_{\text{IG}}\left(\omega \,\middle|\, \frac{b}{|\psi|}, b^2\right) = \frac{b}{\sqrt{2\pi\omega^3}} \exp\left\{-\frac{\psi^2}{2\omega}\left(\omega - \frac{b}{|\psi|}\right)^2\right\}.$$

When $\psi = 0$, the inverse Gaussian density reduces to an inverse gamma density,

$$p_{\text{IGa}}(\omega \mid \tfrac{1}{2}, \tfrac{b^2}{2}) = \frac{b}{\sqrt{2\pi\omega^3}} \exp\left\{-\frac{b^2}{2\omega}\right\}.$$

Finally, the third term we have called $\Phi(\omega \mid b)$,

$$\begin{aligned}\Phi(\omega \mid b) &= \sum_{n=0}^{\infty} (-1)^n \phi_n(\omega \mid b) \\ \phi_n(\omega \mid b) &= \frac{\Gamma(n+b)}{\Gamma(n+1)} \frac{2n+b}{\Gamma(b+1)} \exp\left\{-\frac{2n(n+b)}{\omega}\right\},\end{aligned}$$

where each term, $\phi_n(\omega \mid b)$, is nonnegative, and $\phi_0(\omega \mid b) = 1$.

Figure 8.2a plots $\Phi(\omega \mid b)$ for various values of b. In all cases, it appears that $\Phi(\omega \mid b)$ is monotonically decreasing and its range is $[0, 1]$. We have not proven this, but our numerical experiments suggest that it is true. We formalize this as a conjecture:

Conjecture 1. *For all $b > 0$, $\Phi(\omega \mid b)$ is a monotonically decreasing function of ω with,*

$$\lim_{0 \leftarrow \omega} \Phi(\omega \mid b) = 1,$$
$$\textit{and,} \quad \lim_{\omega \to \infty} \Phi(\omega \mid b) = 0.$$

Assuming this conjecture is true, as our plots suggest, all three terms in (8.10) are nonnegative. With $\Phi(\omega \mid b) \leq 1$, the product $\alpha^{-1}(b, \psi)\, p_{\text{IG}}(\omega \mid \frac{b}{|\psi|}, b^2)$ must dominate $p_{J^*}(\omega \mid b, \psi)$. Thus, the inverse Gaussian is a natural proposal distribution for a rejection sampling algorithm. To determine whether a proposed value of ω is accepted, we must sample $u \sim \text{Unif}(0, 1)$, and check whether $u < \Phi(\omega \mid b)$.

The acceptance probability is $\alpha(b, \psi)$, the inverse of the scaling constant. It is bounded between $[\frac{1}{2}, 1]$ when $b \leq 1$. The lower bound (worst case) is achieved when $\psi = 0$ and $b = 1$. The upper bound (best case) is approached as b goes to zero or $|\psi|$ goes to infinity. This is illustrated in Figure 8.2b for a range of b and ψ. In fact, this rejection sampling algorithm works for $b \geq 1$ as well, but as b increases, the acceptance probability goes to zero. For this regime, the existing approaches of Windle et al. (2014) are a better choice.

Determining acceptance In order to determine whether to accept or reject a proposed value of ω, we need to compare against $\Phi(\omega \mid b)$. This function is not analytically tractable; however, it is still possible to determine whether or not to accept with finite computation. To do so, we use a slight modification of the alternating series method (Devroye, 1986). We exploit the fact that $\Phi(\omega \mid b)$ is an alternating sum, and the terms, $\phi_n(\omega \mid b)$, are eventually monotonically decreasing as a function of the index n for all fixed values of ω and b. We formalize this with the following lemma,

Lemma 1. *For all fixed values of ω and b,*

$$\exists m : \forall n \geq m : \phi_{n+1}(\omega \mid b) < \phi_n(\omega \mid b).$$

Proof. We show that the ratio of ϕ_{n+1} to ϕ_n is a decreasing function whose limit is zero.

$$\begin{aligned}
\frac{\phi_{n+1}(\omega \mid b)}{\phi_n(\omega \mid b)} &= \frac{\Gamma(n+1)\Gamma(n+1+b)(2n+2+b)\exp\left\{-\frac{2(n+1)(n+1+b)}{\omega}\right\}}{\Gamma(n+2)\Gamma(n+b)(2n+b)\exp\left\{-\frac{2n(n+b)}{\omega}\right\}} \\
&= \frac{(n+b)(2n+b+2)}{(n+1)(2n+b)}\exp\left\{-\frac{4n+2b+2}{\omega}\right\} \\
&= \ell(n)r(n),
\end{aligned}$$

where

$$\begin{aligned}
\ell(n) &= \frac{2n^2+nb+2n+b+2(n+1)b+b^2}{2n^2+nb+2n+b} \\
&= 1+\mathcal{O}\left(\tfrac{1}{n}\right), \\
r(n) &= \exp\left\{-\frac{4n+2b+2}{\omega}\right\}.
\end{aligned}$$

Observe that $\ell(n)$ is monotonically decreasing toward one as n approaches infinity. The rate of convergence is inverse polynomial in n. In contrast, $r(n)$ decreases to zero exponentially quickly as n approaches infinity. Thus, there exists a threshold m such that this ratio is less than one for all $n \geq m$. Equivalently,

$$\forall\, n \geq m :\ \phi_{n+1}(\omega \mid b) \leq \phi_n(\omega \mid b).$$

□

Lemma 1 guarantees that once we have computed the increasing terms, all subsequent partial sums for even n are upper bounds, and all subsequent partial sums for odd n are lower bounds on $\Phi(\omega \mid b)$. To determine acceptance of u, we evaluate until we find an upper bound less than u, at which point we reject, or a lower bound greater than u, at which point we accept. In practice, determining acceptance takes only a small number of iterations.

Algorithm 8.4 provides pseudocode for the final rejection sampling algorithm.

Require: $b > 0, \psi \in \mathbb{R}$
  accept $\leftarrow$ False
  while not accept **do**
    $\omega \sim \mathrm{IG}\left(\frac{b}{|\psi/2|}, b^2\right)$ ▷ Inv. Gaussian proposal
    $u \sim \mathrm{Unif}(0, 1)$ ▷ Sample acceptance variable
    $\Phi = 1$ ▷ Initialize partial sum with first term
    for $n = 1$ to ∞ **do**
      $\Phi \leftarrow \Phi + (-1)^n \phi_n(\omega \mid b)$ ▷ Update partial sum
      if $\phi_n(\omega \mid b) < \phi_{n-1}(\omega \mid b)$ **then** ▷ Check if terms are decreasing
        if n odd **and** $u \leq \Phi$ **then** ▷ Compare to lower bound
          accept $\leftarrow$ True ▷ Accept and return
          break
        end if
        if n even **and** $u > \Phi$ **then** ▷ Compare to upper bound
          break ▷ Reject and make new proposal
        end if
      end if
    end for
  end while
  return $\frac{1}{4}\omega$

Algorithm 8.4: A rejection sampling algorithm for the Pólya-gamma distribution that is most efficient in the "small-shape" ($b < 1$) regime.

8.6 Conclusion

This chapter has explored various facets of modeling neural spike trains with switching linear dynamical systems models. This powerful model for nonlinear dynamical systems contains a number of simpler models as special cases. We have shown how a simple MCMC inference algorithm based on the Pólya-gamma augmentation provides a unified means of performing inference for the SLDS and its special cases.

As we consider hierarchical models like these — models constructed out of layers of latent structure — we must turn our attention to the important question of model selection. How should we justify our modeling choices? Marginal likelihood estimates provide one answer to this question. We have shown how the same Pólya-gamma augmentations can be applied inside annealed impor-

tance sampling algorithms, one of the most successful means of approximating marginal likelihoods. In order to make these methods work in practice, however, we needed to improve the efficiency of sampling the Pólya-gamma distribution in the "small shape" regime. By leveraging a particular decomposition of the related Jacobi density, we derived a novel rejection sampling algorithm with acceptance probability of at least one half.

Next, we turn our attention to another important question. For all their structure, what can these models teach us about neural computation? The next chapter provides some initial attempts to connect the methods we have developed thus far to more abstract theoretical models of neural computation.

9

Reverse Engineering Bayesian Computations from Spike Trains

The preceding chapters have developed a range of probabilistic models for neural spike trains that leverage our intuitions about neural types, features, and states to inform structured prior distributions over the dynamics of neural activity. In this chapter, we take a first step toward reconciling these intuitive models with the host of theoretical models of neural computation. From a Bayesian perspective, theoretical models can be seen as prior distributions on activity — albeit highly sophisticated ones. By connecting theory to observation in a hierarchical probabilistic model, we provide the link necessary to test, evaluate, and revise our theories in a data-driven fashion. As an exercise in meta-reasoning, the theory we test with this Bayesian approach is that neural populations are themselves performing Bayesian inference.

This chapter is organized as follows. First, in Section 9.1 we briefly review the "Bayesian brain" hypothesis, the various lines of evidence supporting this hypothesis. While not strictly required by the Bayesian brain hypothesis, we review some of the hypothesized means by which neurons could represent probability distributions and carry out Bayesian calculations. Then, in Section 9.2 we present a distributed representation scheme, and in Section 9.3 we provide a novel analysis of its complexity in the spirit of Valiant (1994). This provides important constraints on biological plausibility of this scheme and the number of neurons we must observe in order to test this theory. Section 9.4 adapts existing theories to show how neural circuits could perform mean-field variational

inference in a restricted class of graphical models given our representation scheme. A simple example of inference in a mixture model is illustrated in Section 9.5. Finally, in Section 9.6 we show that the dynamics of inference in this model are equivalent to a nonlinear autoregressive model with weights drawn from a stochastic block model, thus providing a "top-down," theoretical justification for the intuitive models developed in Chapter 5. With this insight, we show how simple probabilistic models can be reverse engineered from spike trains using Bayesian inference and a Bayesian theory of neural computation. Section 9.7 considers some important open problems and directions for future work.

9.1 The "Bayesian Brain" Hypothesis

Bayesian theories of neural computation address a fundamental question: how do organisms reason, act, and make decisions given only limited, noisy information about the world around them? Bayes' rule tells us how an optimal agent should combine noisy information with prior knowledge to make posterior inferences. That the brain may employ or approximate Bayesian methods is an idea that dates back as far as von Helmholtz and Southall (1925). At the cognitive level, Bayesian models have proven extraordinarily useful for understanding and explaining human and animal behavior (Tenenbaum et al., 2011; Griffiths et al., 2008). These cognitive models span a variety of domains, from lower level systems like visual perception (Knill and Richards, 1996; Brainard and Freeman, 1997; Weiss et al., 2002; Yuille and Kersten, 2006; Stocker and Simoncelli, 2006; Simoncelli, 2009) and motor control (Körding and Wolpert, 2004) to higher level systems of sensory integration (Ernst and Banks, 2002), time interval estimation (Jazayeri and Shadlen, 2010), language processing (Chater and Manning, 2006), attention (Whiteley and Sahani, 2012; Chikkerur et al., 2010; Dayan and Solomon, 2010), and learning (Tenenbaum et al., 2006; Courville et al., 2006) The success of these models in explaining behavior suggests that the brain may be performing, or at least approximating, Bayesian computations.

Further buttressing the Bayesian brain hypothesis, some experiments have shown Bayes-optimal behavior along with simultaneous neural responses that are strongly correlated with relevant probabilistic quantities. For example, Yang and Shadlen (2007) trained monkeys to make an eye movement to either the left or the right based on an observed set of shapes. Each shape contributed an additive "weight" to the log probability that reward would be given for leftward movements rather than rightward, so the optimal strategy (once the weights were learned) was to sum the weights, compute the log probability of left versus right, and choose the direction most likely to yield a re-

ward. In effect, the monkeys had to perform inference in a simple mixture model. The monkeys learned to perform this task optimally, and Yang and Shadlen (2007) found that the firing rates of neurons in parietal cortex were proportional to the log likelihood ratio of left versus right. Subsequent work showed that when the paradigm was extended to allow the monkey to opt-out of making a decision and obtain a smaller but guaranteed reward, the monkey chose to opt out only when the probability of left versus right was below a threshold. This implies that the brain has access to not only the most likely direction, but also its uncertainty (Kiani and Shadlen, 2009).

Further evidence of neural probabilistic inference has been found in other simple tasks. In a time interval reproduction task, non-human primates exhibited behavior consistent with a Bayesian model, and simultaneous recordings in parietal cortex found that some neurons encoded interval estimates that could support this behavior (Jazayeri and Shadlen, 2015). In another line of work, neural correlates of multisensory cue integration were found in macaque monkeys performing a heading discrimination task. These neurons combined both visual and vestibular inputs in a manner consistent with Bayesian theory (Gu et al., 2008; Morgan et al., 2008; Fetsch et al., 2009; 2012). While these experiments provide some compelling evidence in favor of a simple probabilistic computations in neural circuits, there is a large gap between these experiments and the rich array of cognitive phenomena surveyed above. To bridge this gap, we need a broader theory of Bayesian inference in neural circuits, and more powerful tools to link these theories to neural activity.

The past decade has witnessed a surge of interest in theoretical models of Bayesian inference with spiking neurons, and this work has been the subject of a number of recent surveys (Simoncelli, 2009; Fiser et al., 2010; Pouget et al., 2013; Ma and Jazayeri, 2014). These theories can be broadly characterized by their answers to three successive questions:

1. How are probabilities are represented, and how are the conditional probability distributions that constitute the probabilistic model encoded? Are these distributions represented in a parametric manner? How are the parameters instantiated in a neural system?

2. Given a representation, how do neural dynamics compute the desired posterior distribution? In other words, what is the algorithm of probabilistic inference, and how is it reified in a population of neurons? These dynamics must respect the natural constraints of neural systems, for example, that neural connectivity is sparse and that neurons have limited computational power.

3. Finally, how are the parameters of the probabilistic model learned, and how are new variables of interest incorporated into an existing model?

The simplest and most common answer to the first question is that neural firing rates are proportional to probability (Hinton and Sejnowski, 1983; Hinton, 1992; Anderson and Essen, 1994; Barber et al., 2003; Buesing et al., 2011; Berkes et al., 2011; Nessler et al., 2013; Legenstein and Maass, 2014) or some function of the probability, like its log (Rao, 2004; Beck and Pouget, 2007; Rao, 2007; Litvak and Ullman, 2009). Others have suggested that probability distributions are encoded implicitly by the stochasticity of neurons (Zemel et al., 1998; Sahani and Dayan, 2003; Ma et al., 2006). Still others have contended that neurons employ a predictive coding scheme to convey probabilistic information (Rao and Ballard, 1999; Deneve, 2008; Huang and Rao, 2011). According to the predictive coding hypothesis, neurons only communicate spikes when their internal state cannot otherwise be inferred by their downstream neighbors. Finally, an interesting variant of rate coding suggests that distributions may be implicitly encoded in the number of neurons representing a particular value or, similarly, in the width of neural tuning curves (Shi and Griffiths, 2009; Ganguli and Simoncelli, 2010).

Along with this host of representational hypotheses has come an equally broad set of proposed inference algorithms. In the simplest models, like mixture models with a single latent variable, inference can be performed exactly in a single step. For more complicated models, some dynamic and often approximate inference algorithm is necessary. Of these, belief propagation and related message passing algorithms (Rao, 2007; Litvak and Ullman, 2009), variational inference (Friston, 2010; Nessler et al., 2013), and sampling based methods like Markov chain Monte Carlo (MCMC) (Hoyer and Hyvarinen, 2003; Buesing et al., 2011; Berkes et al., 2011; Gershman et al., 2012b; Legenstein and Maass, 2014), Hamiltonian Monte Carlo (Aitchison and Lengyel, 2014), importance sampling (Shi and Griffiths, 2009), and particle filtering (Lee and Mumford, 2003) have all been suggested. This amazing diversity speaks to both the computational power of neural populations as well as the enormous challenge in winnowing the field of contending theories.

The question of learning has received less attention; indeed, many theories have ignored it completely. Those that have addressed it tend to equate learning with synaptic plasticity. In some cases, synaptic plasticity rules like spike-timing dependent plasticity can be seen as maximizing a lower bound on the marginal log likelihood (Friston, 2010; Nessler et al., 2013; Rezende et al., 2011). In others, the stochasticity of synapses (e.g. stochastic vesicle release) is seen as sampling from a distribution over weights (Aitchison and Latham, 2015; Kappel et al., 2015b;a; Tully et al., 2014). These theories address unsupervised learning of model parameters, but the larger question of learning model structure, via either supervised or unsupervised means, remains largely a mystery.

Given the breadth and depth of existing theories of neural inference, our intention here is not to

present a radically novel theory. Instead, our focus is on how we may assess the viability of a theory of neural inference. To that end, we provide a detailed description of a distributed representation of probability, analyze its complexity, show how inference could be performed with this representation, and provide a simple example of how this representation could be reverse engineered from neural spike train recordings.

9.2 A Direct Distributed Representation of Probability Distributions

As described above, the simplest representation of probability is a *direct* representation in which neural firing rates reflect instantaneous probabilities. Assume a population of neurons is responsible for representing the distribution over values that a set of random variables may take on. We denote this set of variables by, $\boldsymbol{z} = \{z_1, \ldots, z_J\}$. For simplicity, assume for now that these variables can only assume a discrete set of values, $\{1, \ldots, K\}$. Our neural population is thus tasked with representing probability vectors, $\boldsymbol{\pi}^{(j)}$, for each variable. The entries of these vectors are, $\pi_k^{(j)} = \Pr(z_j = k)$. In Section 9.7, we discuss how this representation could be extended to continuous probability density functions by assigning each neuron a basis function or tuning curve over the support of the distribution, as in Barber et al. (2003); Ma et al. (2006); Beck and Pouget (2007).

In a *distributed* representation, each variable-value pair, (z_j, k), is associated with a subpopulation of R neurons. This is inspired by Valiant (1994; 2005), and is similar to the ensemble based neural sampling code of Legenstein and Maass (2014). We further assume that these subpopulations are *non-overlapping* such that each neuron can be represented with at most one variable-value pair. Let $j_n \in \{1, \ldots, J\}$ denote the index of the specific variable and $k_n \in \{1, \ldots K\}$ denote the particular value that neuron n represents.

Now let $s_{t,n}$ denote the number of spikes fired by neuron n in the t-th time bin. The relative spike counts encode instantaneous probability distributions for each variable. If neurons representing a particular value fire twice as many spikes as neurons representing a competing value, then the first value is twice as likely. We introduce the notion of an *integration time*, T_I, over which spikes are counted to estimate the probability distribution. With this notation, the empirical distribution encoded by a population at a particular instant in time, $\widehat{\boldsymbol{\pi}}_t^{(j)}$, is defined by,

$$\widehat{\pi}_{t,k}^{(j)} = \frac{\sum_{\Delta=1}^{T_I} \sum_{n=1}^{N} \mathbb{I}[j_n = j, k_n = k]\, s_{t-\Delta,n}}{\sum_{\Delta=1}^{T_I} \sum_{n=1}^{N} \mathbb{I}[j_n = j]\, s_{t-\Delta,n}}.$$

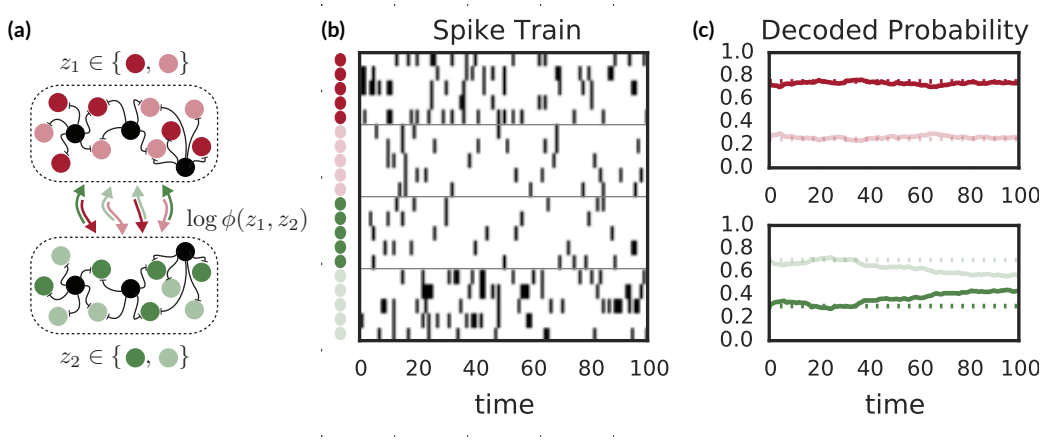

Figure 9.1: Example of a population of neurons encoding marginal probability distributions for two binary random variables, z_1 and z_2. Each variable is represented by a population of neurons, which is further divided into subpopulations for each value the variable can take on. In **(a)**, z_1 is represented by red neurons, with dark red neurons representing $z_1 = 1$ and light red representing $z_1 = 0$. Interneurons (black) provide normalization in the form of local inhibition. Excitatory connections between populations implement probabilistic inference. **(b)** The population spike train encodes the marginal distributions. For example, $\Pr(z_1 = 1)$ is proportional to the spike count of the subpopulation of dark red neurons. **(c)** These probabilities are decoded by integrating spike counts for each subpopulation over time and normalizing across subpopulations for each variable. Dotted lines: true probability.

The numerator counts spikes from neurons representing the particular value, $z_j = k$; the denominator counts all spikes from neurons representing z_j.

We assume these neurons are stochastic, each endowed with an instantaneous firing rate, $\lambda_{t,n}$, which gives rise to an instantaneous spike count, $s_{t,n}$ according to a Poisson distribution, $s_{t,n} \sim \text{Poisson}(\lambda_{t,n})$. Thus, while the firing rate may encode one distribution, the distribution that is read out from a finite number of spikes will differ.

To summarize, the direct distributed representation entails the following assumption:

Assumption 1. *Neurons represent discrete probability distributions with a direct, distributed code. Each variable-value pair is allotted R neurons that emit spikes according to a Poisson distribution. Spikes are integrated over the subpopulation and over an integration time window, T_I, to obtain an unnormalized probability for the variable-value pair. By normalizing across subpopulations, they decode the probability distribution.*

9.3 Complexity of the Direct Distributed Representation

In pioneering work, Valiant (1994) suggested that the tools of computational learning theory, which provide rigorous computational limits on statistical learning, may apply equally well to learning in biological systems. All that is needed is a model of biological computation and a precisely stated learning algorithm.* However, if the resources required by this algorithm (e.g. number of neurons or spikes) grow exponentially quickly with the size of learning problem, then it is highly unlikely that the algorithm is employed by the brain. In other words, biology is bound by the same limits of computational tractability as our silicon devices, but the natural measure of complexity may be spikes and synapses rather than FLOPs and bytes.

Valiant (1994) laid the groundwork for a model of neural computation that emphasizes a few key features: distributed representations, and sparse, random connectivity. In this and following work (Valiant, 2005; 2006), he showed that simple tasks, like learning an association between two variables and learning a linear classifier, could be performed efficiently within this model. Since the problem of approximate Bayesian inference is NP-hard in the worst case (Dagum and Luby, 1993; Roth, 1996), a provably tractable algorithm for neural inference is out of the question. Instead, we focus on the complexity of a prerequisite problem: simply representing a distribution with a population of stochastic neurons. If this is tractable, then we may consider the question of inference and the scenarios in which it may be possible.

Interestingly, the question of complexity has recently arisen in a somewhat different context. Gao and Ganguli (2015) have developed a theory of "task complexity" that predicts the dimensionality of neuronal dynamics as well as the number of neurons that must be measured in order to accurately recover the dynamics. In their case, the quantities of interest are the total number of neurons, number of observed neurons, number of stimuli, length of recording, and the unknown dimensionality of neural data. By fixing some of these parameters, they obtain bounds on the remaining parameters that govern when the dimensionality can be recovered. In our case, we derive similar results that govern the accuracy with which an encoded probability distribution can be recovered, either by a downstream population of neurons, or by an experimental observer.

In our case, we are concerned with the complexity, in terms of the number of neurons, time steps, or spikes, of stochastically encoding a distribution such that the decoded distribution is, with high confidence, within some tolerable error of the true distribution. The stochasticity of the spike

* Of course, the difficulty lies in specifying such a model and algorithm! While this is surely a challenge, there is no substitute for a formal declaration of assumptions.

counts implies that at any instant in time, the probability distribution that is represented by the population will be a random variable. First, we will show that this representation is unbiased. That is, if the firing rates, $\lambda_{t,n}$, are proportional to the probability, $\pi_{k_n}^{(j_n)}$, then the empirical probability distribution, $\widehat{\boldsymbol{\pi}}_t^{(j)}$, will equal $\boldsymbol{\pi}^{(j)}$ in expectation. Since we will be focusing on the representation of a single random variable z_j, we drop the superscripts $^{(j)}$ and $^{(j_n)}$ for the remainder of this section.

Lemma 2. *If the firing rates are proportional to a given probability distribution over the integration time window then the probability distribution represented by the population will have expectation equal to the given probability distribution. That is, if $\lambda_{t,n} = \lambda_{\mathsf{max}} \pi_{k_n}$, then $\mathbb{E}[\widehat{\boldsymbol{\pi}}_t] = \boldsymbol{\pi}$.*

Proof. Let,

$$S_{t,k} = \sum_{\Delta=1}^{T_I} \sum_{n=1}^{N} \mathbb{I}[j_n = j, k_n = k]\, s_{t-\Delta,n},$$

and

$$S_t = \sum_{\Delta=1}^{T_I} \sum_{n=1}^{N} \mathbb{I}[j_n = j]\, s_{t,n} = \sum_{k=1}^{K} S_{t,k}.$$

Iterating expectations, we have,

$$\mathbb{E}[\widehat{\boldsymbol{\pi}}_t] = \mathbb{E}\left[\frac{1}{S_t}\left(S_{t,1}, \ldots, S_{t,K}\right)\right] = \mathbb{E}_{S_t}\left[\mathbb{E}_{\left(S_{t,1},\ldots,S_{t,K}\right)}\left[\frac{1}{S_t}\left(S_{t,1}, \ldots, S_{t,K}\right) \,\middle|\, S_t\right]\right].$$

Since $s_{t,n}$ are independent Poisson random variables, their partial sums are as well. Specifically,

$$\begin{aligned} S_{t,k} &\sim \text{Poisson}\left(\sum_{\Delta=1}^{T_I} \sum_{n} \mathbb{I}[j_n = j, k_n = k] \lambda_{t,n}\right) \\ &= \text{Poisson}\left(\lambda_{\mathsf{max}} T_I \sum_{n} \mathbb{I}[j_n = j, k_n = k]\, \pi_k\right) \\ &= \text{Poisson}(\lambda_{\mathsf{max}}\, T_I\, R\, \pi_k), \end{aligned} \tag{9.1}$$

which implies,

$$S_t = \sum_k S_{t,k} \sim \text{Poisson}(\lambda_{\mathsf{max}}\, T_I\, R).$$

Moreover, by the Poisson superposition principle, the vector $(S_{t,1}, \ldots, S_{t,K})$ is multinomial distributed given S_t,

$$\begin{aligned}(S_{t,1}, \ldots, S_{t,K}) \mid S_t &\sim \text{Mult}\left(S_t, \left(\frac{\lambda_{\mathsf{max}}\, T_I\, R\, \pi_1}{\lambda_{\mathsf{max}}\, T_I\, R}, \ldots, \frac{\lambda_{\mathsf{max}}\, T_I\, R\, \pi_K}{\lambda_{\mathsf{max}}\, T_I\, R}\right)\right) \\ &= \text{Mult}(S_t, \boldsymbol{\pi}),\end{aligned}$$

with expectation $\boldsymbol{\pi} S_t$. Plugging this into the iterated expectation above,

$$\begin{aligned}\mathbb{E}[\widehat{\boldsymbol{\pi}}_t] &= \mathbb{E}_{S_t}\left[\mathbb{E}_{(S_{t,1},\ldots,S_{t,K})}\left[\frac{1}{S_t}\left(S_{t,1}, \ldots, S_{t,K}\right) \,\middle|\, S_t\right]\right] \\ &= \mathbb{E}_{S_t}\left[\frac{\boldsymbol{\pi} S_t}{S_t}\right] \\ &= \boldsymbol{\pi}.\end{aligned}$$

Thus, this stochastic encoding is unbiased. □

While this stochastic representation may have the correct expectation, we would like to characterize the probability that it is "close" to its mean. As we hypothesized above, the difference between the true probability and that represented by the population should shrink as the number of spikes grows. We measure this difference with the ℓ_∞ norm of the difference between two probability vectors,

$$||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty = \max_k |\widehat{\pi}_{t,k} - \pi_k|.$$

While somewhat unorthodox, this metric is similar in spirit to the total variation distance. The following theorem provides an upper bound on the number of spikes required to guarantee that the represented probability differs from the true probability by more than ϵ.

Theorem 1. *Given a fixed probability vector $\boldsymbol{\pi}$, firing rates $\lambda_{t,n} = \lambda_{\mathsf{max}}\pi_{k_n}$ over the integration time window, a fixed error level $\epsilon < 1$, and a desired confidence $\delta < 1$, there exists a minimum number of spikes S^* such that if $S_t \geq S^*$, the conditional probability of error is bounded by $\Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon \mid S_t) < \delta$. Furthermore, this minimum number of spikes is at most,*

$$S^* \leq \frac{1}{2\epsilon^2} \ln \frac{2K}{\delta},$$

Proof. First, consider the probability that a particular entry differs from its mean by more than ϵ.

$$\begin{aligned}
&\Pr(|\widehat{\pi}_k - \pi_k| > \epsilon \,|\, S_t) \\
&\quad = \Pr(\widehat{\pi}_k - \pi_k > \epsilon \,|\, S_t) + \Pr(\widehat{\pi}_k - \pi_k < -\epsilon \,|\, S_t) \\
&\quad = \Pr\left(S_{t,k} > S_t \pi_k \left(1 + \frac{\epsilon}{\pi_k}\right) \,\middle|\, S_t\right) + \Pr\left(S_{t,k} < S_t \pi_k \left(1 - \frac{\epsilon}{\pi_k}\right) \,\middle|\, S_t\right) \\
&\quad = \Pr\left(S_{t,k} > \mathbb{E}[S_{t,k} \,|\, S_t] \left(1 + \frac{\epsilon}{\pi_k}\right)\right) + \Pr\left(S_{t,k} < \mathbb{E}[S_{t,k} \,|\, S_t] \left(1 - \frac{\epsilon}{\pi_k}\right)\right)
\end{aligned}$$

As in Lemma 2, we have used the fact that $S_{t,k} \,|\, S_t \sim \mathrm{Bin}(S_t, \pi_k)$ and hence has expectation $S_t \pi_k$. The probability of this binomial random variable exceeding its mean by a multiplicative constant is a decreasing function of the number of spikes, S_t. This implies that there exists a minimum number of trials S^* such that for $S_t \geq S^*$, this probability of error is bounded above by δ, hence proving the first part of the theorem.

Now suppose $S_t = S^*$. We use a Chernoff bound to upper bound the probability that the binomial random variable, $S_{t,k}$, deviates from its mean by more than a multiplicative factor. Leveraging the fact that $\pi_k \leq 1$, we have,

$$\begin{aligned}
\Pr\left(S_{t,k} > \mathbb{E}[S_{t,k} \,|\, S^*] \left(1 + \frac{\epsilon}{\pi_k}\right)\right) &\leq \exp\left\{-2S^*\epsilon^2\right\}, \\
\Pr\left(S_{t,k} < \mathbb{E}[S_{t,k} \,|\, S^*] \left(1 - \frac{\epsilon}{\pi_k}\right)\right) &\leq \exp\left\{-2S^*\epsilon^2\right\},
\end{aligned}$$

which together imply,

$$\Pr(|\widehat{\pi}_{t,k} - \pi_k| > \epsilon \,|\, S^*) \leq 2\exp\left\{-2S^*\epsilon^2\right\}.$$

We bound the maximum deviation of any entry in $\widehat{\boldsymbol{\pi}}$ with a union bound,

$$\Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon \,|\, S^*) \leq 2K\exp\left\{-2S^*\epsilon^2\right\}.$$

Setting this probability equal to δ yields the desired bound on S^*,

$$S^* \leq \frac{1}{2\epsilon^2} \ln \frac{2K}{\delta}.$$

□

This theorem provides an upper bound on the minimum number of spikes necessary to guarantee that the ℓ_∞-distance between the true and estimated probability vectors is less than ϵ with probability $1-\delta$. Notably, the relevant quantity is the number of spikes S_t, rather than the number of neurons. Thus, there is some flexibility in how the probability is estimated: a small population of neurons could be measured over many time bins, or a large population could be measured over a single time bin. Moreover, the population gain, λ_{max}, could be varied to adjust the number of spikes per time bin.

In practice, the number of spikes cannot be set directly. It, is a Poisson random variable whose mean, from Eq. 9.1, is $\mathbb{E}[S_t] = \lambda_{\mathsf{max}} T_I R$: the expected number of spikes per neuron times the number of neurons per outcome. This leads to the following theorem, which specifies a upper bound on the gain and number of neurons required to guarantee that the ℓ_∞-distance is less than ϵ with probability $1-\delta$.

Theorem 2. *Given a fixed probability vector $\boldsymbol{\pi}$, firing rates $\lambda_{t,n} = \lambda_{\mathsf{max}} \pi_{k_n}$, a fixed error level $\epsilon < 1$, and a desired confidence $\delta < 1$, the probability of error is bounded by $\Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon) < \delta$ if $\lambda_{\mathsf{max}} T_I R \geq \mu^*$, where μ^* is at most,*

$$\mu^* \leq \frac{1}{1-e^{-2\epsilon^2}} \ln \frac{2K}{\delta}.$$

Proof. We have,

$$\begin{aligned}
\Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon) &= \sum_{m=0}^{\infty} \Pr(S_t = m) \Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon \,|\, S_t = m) \\
&\leq \sum_{m=0}^{\infty} \Pr(S_t = m) \times 2K \exp\left\{-2m\epsilon^2\right\} \\
&= 2K \mathbb{E}_{S_t}\left[\exp\left\{-2S_t\epsilon^2\right\}\right] \\
&= 2K \exp\left\{\mu^*(e^{-2\epsilon^2} - 1)\right\},
\end{aligned}$$

where the last line follows from moment generating function of $S_t \sim \text{Poisson}(\mu^*)$. Setting this equal to δ and solving for μ^* yields the stated bound. □

So far we have considered the estimated probability distribution obtained by "reading out" the entire population of neurons. What if we only observe a fraction of the population, as a neuron in a downstream population might? Assume each neuron in the population is "observed" with

probability ρ. The expected number of observed neurons for a given variable-value pair is $R\rho$, and if we see exactly the expected number of neurons for each value (assume it is an integer), the estimated probability distribution will have the correct expectation. However, in practice we will incur some bias from seeing a different number of neurons for each value. Bounding the error theoretically is challenging due to this additional source of randomness, so we instead consider the simple case in which we see exactly $R\rho$ neurons for each value. Then, following the same logic as above, we have the following corollary.

Corollary 1. *Given a fixed probability vector $\boldsymbol{\pi}$, firing rates $\lambda_{t,n} = \lambda_{\mathsf{max}}\pi_{k_n}$ over the integration time window, a fixed error level $\epsilon < 1$, and a desired confidence $\delta < 1$, and ρR observed neurons for each of the K values, the probability of error is bounded by $\Pr(||\widehat{\boldsymbol{\pi}}_t - \boldsymbol{\pi}||_\infty > \epsilon) < \delta$ if*

$$(\lambda_{\mathsf{max}} T_I)(\rho R) \geq \mu^*,$$

where μ^* is at most,

$$\mu^* \leq \frac{1}{1 - e^{-2\epsilon^2}} \ln \frac{2K}{\delta}.$$

Proof. This follows directly from Theorem 2 with ρR substituted for R. □

Corollary 1 provides theoretical connection between the fidelity of the representation, measured in terms of the error ϵ and confidence δ, for a given domain size K, maximum firing rate λ_{max}, integration time T_I, connection probability ρ, and representation size R. The expected spike count is the product of the effective number of neurons, ρR, and the expected number of spikes per neuron, $\lambda_{\mathsf{max}} T_I$. Together, these allow us to deduce a manifold of trade-offs between population size and integration time that will achieve a desired error level and confidence.

Figure 9.2 plots these theoretical bounds. Fig. 9.2a shows the theoretical upper bound on the 95th percentile of the ℓ_∞-distance as a function of the expected spike count for a range of distribution sizes, K. Fig. 9.2b illustrates the trade-offs between effective number of neurons and expected number of spikes per neuron necessary to achieve a desired expected spike count.

Figure 9.3 shows the results of an empirical assessment of this theory under a variety of parameter regimes. For each regime, we present results for discrete distributions over $K = 2$ values (left column) and $K = 10$ values (right column). We sample 1000 discrete distributions from a Dirichlet prior, $\boldsymbol{\pi} \sim \mathrm{Dir}(\mathbf{1}_K)$, and then we encode each true distribution with Poisson spiking neurons and measure the ℓ_∞-distance between the true and encoded distributions.

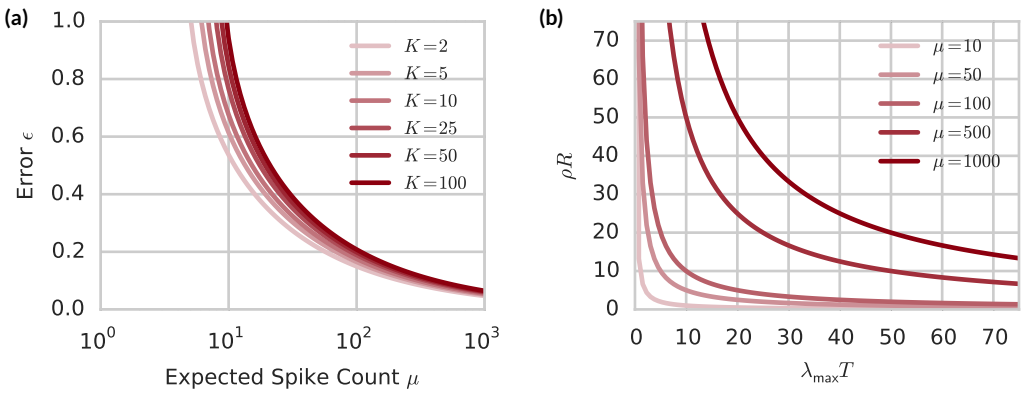

Figure 9.2: Theoretical relationship between between ℓ_∞-distance, expected spike count, and physiological parameters. **(a)** Theoretical upper bound on the 95th percentile of the ℓ_∞-distance, ϵ, as a function of the expected spike count, $\mu = (\rho R)(\lambda_{\text{max}} T_I)$, for increasing values of K. **(b)** The expected spike count is the product of the effective number of neurons, ρR, and the effective number of spikes per neuron, $\lambda_{\text{max}} T_I$. This shows how time and number of neurons can be balanced to obtain the desired expected spike count.

In the top row, we consider the case where the total population spike count is set explicitly, as in Theorem 1. In this case, the spikes are attributed to each value according to a multinomial distribution. We plot the theoretical bound from Theorem 1 in yellow.

In the middle row we measure the error as a function of the representation size, R, for $\rho = 1.0$, under the assumption that spikes are counted in one millisecond bins for $T_I = 10$ milliseconds, and a maximum firing rate of 100Hz (i.e. $\lambda_{\text{max}} = 0.1$). Thus, in expectation the population will emit R spikes, allowing the top and middle rows to be directly compared. That is, in the top row, the population fires exactly the number of spikes expected in the middle row. We see that the stochastic population spike counts does indeed introduce extra variability in the error, as predicted by Theorem 2, though the median error is not substantially different from that of the fixed-S case.

Finally, the bottom row shows the empirical and predicted error as we vary the observation probability, ρ. Here, the representation size is fixed to $R = 25$, and the gain and integration time are set as in the middle row. While a strict upper bound is difficult to derive theoretically, the approximation from Corollary 1 provides a reasonable approximation for this parameter regime.

These complexity-theoretic bounds relate the number of spikes to the distance between the true and estimated distributions. From the number of spikes, we can deduce constraints on the representation size, integration time, and connection probability, for realistic gain levels. While the theoretical bounds do not exactly match the empirical error distributions, they appear to be roughly correct up to a multiplicative factor. Thus, if we can estimate some biophysical properties like con-

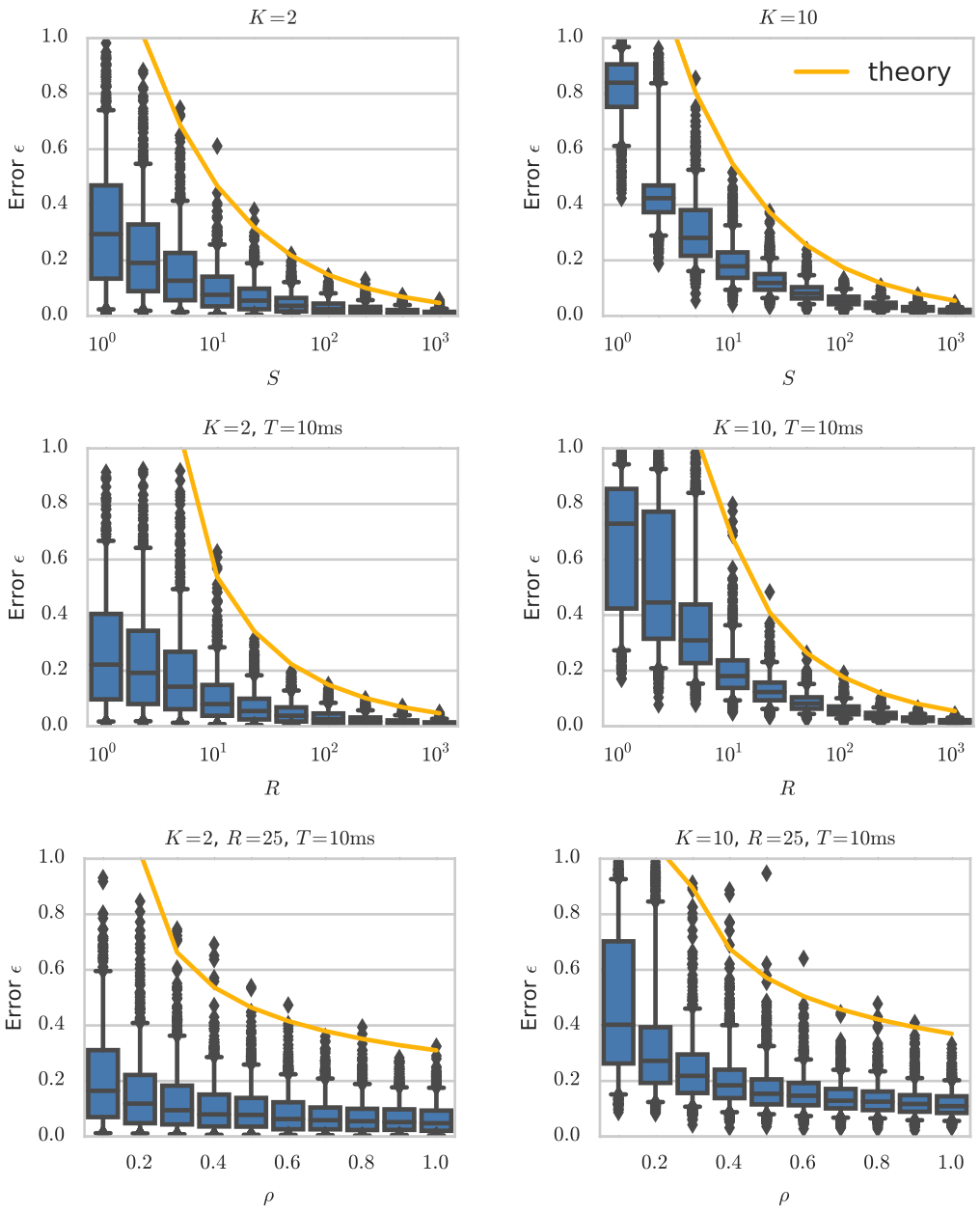

Figure 9.3: Empirical and theoretical ℓ_∞-distance under a variety of parameter regimes. Whiskers show the empirical 5th and 95th percentiles. Yellow line shows the theoretical upper bound on the 95th percentile. **Left** column: $K = 2$. **Right** column: $K = 10$. **Top** row: fixed population spike count, S. **Middle** row: varying the number of neurons per variable. **Bottom** row: varying the connection probability. See text for full description.

nection probabilities, integration times, and firing rates, and we can constrain the error tolerances of the algorithms that consume these probability estimates, then we can estimate the number of neurons that must represent each variable-value pair. This will prove useful in guiding our bottom-up search, and serve as an important constraint for assessing the viability of a direct representation of probability.

9.4 Bayesian Inference with Neural Dynamics

Two forces cause the encoded distributions to change over time. As we interact with the world and receive new inputs, the probabilities of variables change to reflect the new observations. Moreover, even for a fixed set of observed variable assignments, the probabilities of latent variables will change as we perform inference. We show how a simple, iterative inference algorithm can be implemented with biologically plausible neural dynamics.

We assume that as an organism receives new inputs from the world, it updates its posterior distribution over the values of latent variables. Doing so requires a probabilistic model that relates hidden and observed variables via a joint probability distribution. Whereas in previous chapters we have considered directed graphical models, here we assume that the probabilistic models implemented in the brain are best described in terms of a *factor graph*,

$$p(\boldsymbol{z} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{j \in \mathcal{G}} \phi(z_j \mid \boldsymbol{\theta}) \prod_{i,j \in \mathcal{G}} \phi(z_i, z_j \mid \boldsymbol{\theta}) \tag{9.2}$$

The graph, $\mathcal{G}$, specifies unary and pairwise probabilistic dependencies between variables. Each unary factor, $\phi(\cdot \mid \boldsymbol{\theta})$ is a function that maps a variable assignment to a nonnegative real number, and each pairwise factor, $\phi(\cdot, \cdot \mid \boldsymbol{\theta})$, is a function that maps a pair of assignments, say $(z_i = k, z_j = k')$ to a nonnegative real number. The normalizing constant, $Z(\boldsymbol{\theta})$, ensures that the joint probability distribution sums to one. The probabilistic model in Eq. 9.2 reflects a specific assumption about the types of dependency structures neural populations can represent.

Assumption 2. *Neural populations perform inference in probabilistic models that factor into the product of unary and pairwise dependencies.*

A general probabilistic model need not factor into pairwise terms. It may instead have factors that relate three or more latent variables. As we will see, unary and pairwise factors map naturally onto neural biases and synaptic weights. In our proposed neural implementation, higher order fac-

tors would require the interaction of three or more neurons. While this may be realized with dendritic computation or interneurons, these more sophisticated implementations are beyond the scope of this chapter.

In general, the posterior distribution of a subset of hidden variables $\boldsymbol{z}_H \subseteq \boldsymbol{z}$ given the observed variables $\boldsymbol{z}_O = \boldsymbol{z} \setminus \boldsymbol{z}_H$ is,

$$p(\boldsymbol{z}_H \mid \boldsymbol{z}_O, \boldsymbol{\theta}) = \frac{p(\boldsymbol{z}_H, \boldsymbol{z}_O \mid \boldsymbol{\theta})}{\sum_{\boldsymbol{z}_H} p(\boldsymbol{z}_H, \boldsymbol{z}_O \mid \boldsymbol{\theta})}.$$

Typically, this cannot be efficiently computed since it requires a sum over all possible hidden variable assignments. However, as we have seen in previous chapters, there are many methods of approximating posterior distributions. Mean field variational inference maps particularly nicely onto the natural constraints of neural dynamics. In mean field variational inference, the intractable exact posterior distribution is approximated with a tractable, factorized distribution,

$$p(\boldsymbol{z}_H \mid \boldsymbol{z}_O, \boldsymbol{\theta}) \approx q(\boldsymbol{z}_H) \equiv \prod_{z_j \in \boldsymbol{z}_H} q(z_j).$$

The terms in this product are called *variational factors*. We solve for the variational factors that minimize KL-divergence between the true and approximate posterior, $\mathrm{KL}(q(\boldsymbol{z}_H) \,||\, p(\boldsymbol{z}_H \mid \boldsymbol{z}_O, \boldsymbol{\theta}))$. In minimizing the KL-divergence, we simultaneously maximize a lower bound on the log marginal likelihood, $\log p(\boldsymbol{z}_O \mid \boldsymbol{\theta})$.

The simplest method of minimizing this objective is via coordinate descent, iteratively updating the probability of one hidden variable given the probabilities of the rest. Since our variational distribution is factorized, the variational factor for variable z_j must satisfy the mean field consistency equation:

$$\log q(z_j) \simeq \mathbb{E}_{q(\boldsymbol{z}_{\neg j})} \left[\log p(\boldsymbol{z}_H, \boldsymbol{z}_O \mid \boldsymbol{\theta}) \right], \tag{9.3}$$

where $\simeq$ denotes equality up to an additive constant and the expectations are taken with respect to the variational distribution over other hidden variables,

$$q(\boldsymbol{z}_{\neg j}) = \prod_{i \neq j} q(z_i).$$

The additive constant ensures normalization of the probabilities, and will be discussed subse-

quently.

For discrete random variables, the variational factors are simply vectors specifying the posterior probability of each variable, z_j. Under the direct representation described above, the instantaneous values of these factors are encoded in the relative spike counts of populations of neurons,

$$q_t(z_j = k) = \widehat{\pi}_{t,k}^{(j)}$$

To perform inference, the neuronal dynamics must be such that at each time step, the relative spike counts satisfy Eq. 9.3. Explicitly writing the additive constant, $-\log \nu_t^{(j)}$, we have,

$$\begin{aligned}
\log \widehat{\pi}_{t,k}^{(j)} &= -\log \nu_t^{(j)} + \log \phi(z_j = k \,|\, \boldsymbol{\theta}) \\
&\qquad + \mathbb{E}_{q_{t-1}(\boldsymbol{z}_{\neg j})}\left[\sum_{i \in \mathsf{ne}(j)} \log \phi(z_i, z_j = k \,|\, \boldsymbol{\theta})\right] \\
&= -\log \nu_t^{(j)} + \log \phi(z_j = k \,|\, \boldsymbol{\theta}) \\
&\qquad + \sum_{i \in \mathsf{ne}(j)} \sum_{k'=1}^{K} \left[\log \phi(z_i = k', z_j = k \,|\, \boldsymbol{\theta}) \cdot \widehat{\pi}_{t-1,k'}^{(i)}\right] \qquad (9.4) \\
&= -\log \nu_t^{(j)} + \psi_{t,k}^{(j)},
\end{aligned}$$

where

$$\psi_{t,k}^{(j)} = \log \phi(z_j = k \,|\, \boldsymbol{\theta}) + \sum_{i \in \mathsf{ne}(j)} \sum_{k'=1}^{K} \log \phi(z_i = k', z_j = k \,|\, \boldsymbol{\theta}) \cdot \widehat{\pi}_{t-1,k'}^{(i)}.$$

Since $\widehat{\boldsymbol{\pi}}_t^{(j)}$ is a probability distribution, the additive constant must be set to it is normalized. Thus,

$$\begin{aligned}
\widehat{\pi}_{t,k}^{(j)} &= \exp\left\{\psi_{t,k}^{(j)} - \log \nu_t^{(j)}\right\} \\
\implies \nu_t^{(j)} &= \sum_{k'} \exp\left\{\psi_{t,k'}^{(j)}\right\}.
\end{aligned}$$

Now that we have derived theoretically exact mean field updates, we must show how they can be approximated with plausible neural dynamics. We assume that inference occurs on a characteristic time scale of T_I time steps. This reflects the window of time over which neurons estimate proba-

bility distributions. From Lemma 2, we know that if the firing rates of neurons the variable-value pair (z_j, k) are proportional to $\widehat{\pi}_{t,k}^{(j)}$, then in expectation, the empirical distribution represented by the spike counts will be equal to the desired distribution. Thus, we aim to set,

$$\lambda_{t,n} = \lambda_{\mathsf{max}}\, \widehat{\pi}_{t,k_n}^{(j_n)} = \lambda_{\mathsf{max}} \exp\left\{\psi_{t,k_n}^{(j_n)} - \log \nu_t^{(j_n)}\right\},$$

for maximum firing rate, λ_{max}. While this rate function is nearly a linear-nonlinear cascade, as we studied in previous chapters, there is one major impediment to realizing this calculation in biological neurons. Specifically, to compute the activation, a neuron must have access to the *normalized* probabilities of other hidden and visible variables. In practice, a neuron only observes the spike counts of the neurons it receives input from. However, these can be used to estimate the desired probabilities. This motivates our next assumptions,

Assumption 3. *Neurons are sparsely connected to one another. For each ordered pair of neurons, (m, n), the variable $a_{m\to n} \in \{0, 1\}$ indicates whether or not there exists a synaptic connection from neuron m to neuron n. These connections are modeled as independent and identically distributed Bernoulli random variables,*

$$a_{m\to n} \sim \mathrm{Bern}(\rho).$$

We combine these variables into a binary adjacency matrix, $\boldsymbol{A} \in \{0, 1\}^{N\times N}$.

Assumption 4. *All neurons in the population share the same gain, λ_{max}. Thus, neuron n's estimate of $\boldsymbol{\pi}^{(j)}$ is informed by $(\lambda_{\mathsf{max}} T_I)(\rho R)$ spikes, in expectation:*

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{A},\boldsymbol{s}}\left[\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N} \mathbb{I}[j_m = j]\, a_{m\to n}\cdot s_{t-\Delta,m}\right] \\
&\qquad = \mathbb{E}_{\boldsymbol{A},\boldsymbol{s}}\left[\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N}\sum_{k=1}^{K} \mathbb{I}[j_m = j, k_m = k]\, a_{m\to n}\cdot s_{t,m}\right] \\
&\qquad = \lambda_{\mathsf{max}} T_I \rho \sum_{m=1}^{N}\sum_{k=1}^{K} \mathbb{I}[j_m = j, k_m = k]\, \widehat{\pi}_{t,k}^{(j)} \\
&\qquad = (\lambda_{\mathsf{max}} T_I)(\rho R).
\end{aligned}$$

Moreover, the instantaneous probability is well-approximated by,

$$\begin{aligned}\widehat{\pi}_{t,k}^{(j)} &= \frac{\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N}\mathbb{I}[j_m = j, k_m = k]\, a_{m\to n}\cdot s_{t-\Delta,m}}{\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N}\mathbb{I}[j_m = j]\, a_{m\to n}\cdot s_{t-\Delta,m}} \\ &\approx (\lambda_{\mathsf{max}} T_I \rho R)^{-1}\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N}\mathbb{I}[j_m = j, k_m = k]\, a_{m\to n}\cdot s_{t-\Delta,m}.\end{aligned}$$

In other words, the total spike count is concentrated around its mean.

Under the assumption of shared gain, the desired dynamics in Eq. 9.4 simplify to,

$$\lambda_{t,n} = \lambda_{\mathsf{max}} \exp\left\{ b_n + (\lambda_{\mathsf{max}} T_I \rho R)^{-1}\sum_{\Delta=1}^{T_I}\sum_{m=1}^{N} a_{m\to n}\cdot w_{m\to n}\cdot s_{t-\Delta,m} - \log \nu_t^{(j_n)}\right\} \quad (9.5)$$

where

$$b_n = \log \phi(z_{j_n} = k_n \,|\, \boldsymbol{\theta}),$$

and

$$w_{m\to n} = \begin{cases}\log \phi(z_{j_n} = k_n, z_{j_m} = k_m \,|\, \boldsymbol{\theta}) & \text{if } j_m \in \mathsf{ne}(j_n) \\ 0 & \text{o.w.}\end{cases}$$

Thus, the theory provides a normative interpretation of synaptic weights: they reflect the conditional log probabilities for the variable-value pairs represented by the pre- and post-synaptic neurons.

The last step is to compute the normalizing input, $\nu_t^{(j)}$. This requires summing the instantaneous rates of all neurons representing the random variable, z_j. While this is clearly implausible, we may derive a gain controller from an alternative perspective. Normalizing the probability distribution ensures that the expected spike count at any time step for neurons representing z_j is equal to $\lambda_{\mathsf{max}} R$. If the distribution is not properly normalized, the expected spike count will deviate. Thus, a reasonable gain controller can estimate the population can estimate the population rate,

$$\widehat{\lambda}_t^{(j)} = \sum_{\Delta=1}^{T_G}\sum_{n=1}^{N}\mathbb{I}[j_n = j] s_{t-\Delta,n},$$

and set the control input to,

$$\nu_t^{(j)} = \frac{\widehat{\lambda}_t^{(j)}}{\lambda_{\mathsf{max}} T_G R}.$$

The time scale of the gain controller is typically less than the time scale of inference, that is $T_G < T_I$.

Having shown that variational inference is theoretically plausible, we consider a simple example.

9.5 Example of a Simple Mixture Model

Consider a simple mixture model with a single latent variable denoting the mixture component, $z \in \{1, \ldots, K\}$, and a set of conditionally independent observations, $\{x_j\}_{j=1}^J$. In this example, we let the observations be Bernoulli random variables. The model is parameterized by a marginal class probability vector, $\boldsymbol{\alpha}$, which we assume is uniform, and class-conditional probabilities for each observation, $p_{j,k} = \Pr(x_j = 1 \mid z = k)$. Together, these specify the probabilistic model,

$$\begin{aligned} \boldsymbol{\alpha} &= \tfrac{1}{K}\mathbf{1}, & z &\sim \text{Discrete}(\boldsymbol{\alpha}), \\ p_{j,k} &\sim \text{Beta}(\tfrac{1}{2}, \tfrac{1}{2}), & x_j &\sim \text{Bern}(p_{j,z}). \end{aligned}$$

This corresponds to a factor graph with,

$$\begin{aligned} \phi(z = k) &= \alpha_k, \\ \phi(x_j = 1, z = k) &= p_{j,k}, \\ \phi(x_j = 0, z = k) &= 1 - p_{j,k}. \end{aligned}$$

We simulate inference in this model with a population of neurons. Each variable-value pair is represented by $R = 10$ neurons, and each pair of neurons is connected with probability $\rho = 0.5$. The maximum firing rate is set to $\lambda_{\mathsf{max}} = 100$Hz, and the integration time window is set to $T_I = 100$ms. Hence, the expected spike count used to estimate probabilities is $(\rho R)(\lambda_{\mathsf{max}} T_I) = 50$ spikes. The neurons representing the observed variable assignment $x_j = k$ are externally driven at a rate of $0.95\lambda_{\mathsf{max}}$ if $x_j = k$, and a rate of $0.05\lambda_{\mathsf{max}}$ if $x_j \neq k$. We assume that the synaptic weights have already been learned and reflect the exact log of the pairwise factors.

Figure 9.4 illustrates inference dynamics for a changing stimulus. Every second, the observed variables are driven with a new pattern, which leads to a new posterior distribution over the latent variable. With each change in input, the neurons representing the latent variable adjust their firing

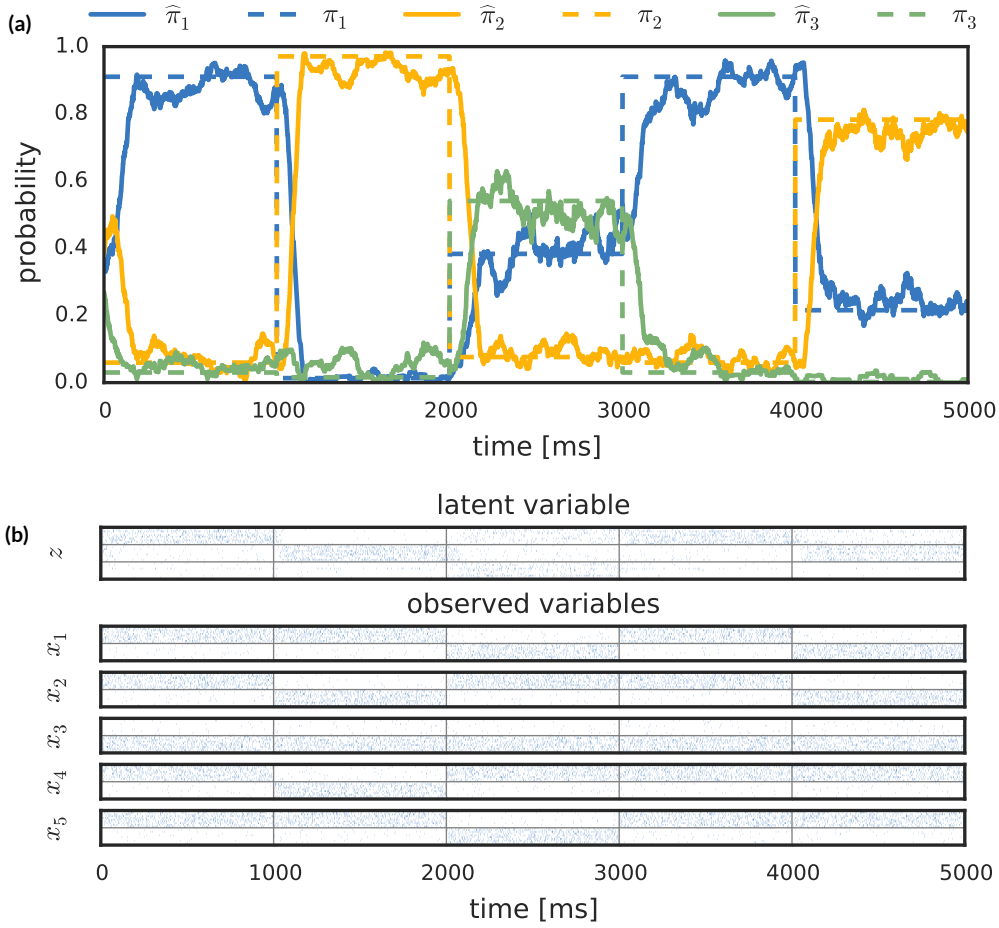

Figure 9.4: Example of neural inference a simple mixture model with one latent variable, $z \in \{1, 2, 3\}$, indicating which of the three mixture components gave rise to the data. The observations consist of 5 conditionally independent binary variables, $x_1, \ldots, x_5$, whose values change every second. **(a)** The empirical probabilities (solid lines) decoded from the spike train, and the true posterior (dashed line). Stochasticity arises from noisy inputs and Poisson spike counts. **(b)** The underlying spike trains of the neurons representing z and x_i. Horizontal gray lines distinguish subpopulations of $R = 10$ neurons for each value; vertical lines denote times of stimulus change.

rates to reflect this new probability. Fig. 9.4a shows the decoded probabilities over time (solid lines) along with the true posterior (dashed lines). Despite many sources of stochasticity, the decoded probabilities do converge to the correct posterior values. The 100ms integration time is reflected in the delayed convergence upon each change in stimulus. Fig. 9.4b shows the spike trains from which these probabilities were decoded. The neurons are ordered according to the value they encode and the subpopulations of R neurons are separated by horizontal light gray lines. Vertical lines indicate

changes in input. Overall, the neurons fire at between 30 and 40Hz, with a dynamic range of about 0 to 100Hz, as expected.

9.6 Reverse Engineering the Probabilistic Model from Spike Trains

Given this "top-down" theory of neural computation, can we reverse engineer the probabilistic model from neural recordings? To do so, we need to infer the subpopulations of neurons that encode each variable-value pair, as well as the characteristic weights that connect each subpopulation. We show that this is possible using the generalized linear models and structured network priors described in Chapter 5.

Recall the theoretical dynamics proposed in Section 9.4, reproduced here in slightly simplified form,

$$\lambda_{t,n} = \lambda_{\mathsf{max}} \exp\left\{ b_n + \gamma \sum_{\Delta=1}^{T_I} \sum_{m=1}^{N} a_{m\to n} \cdot w_{m\to n} \cdot s_{t-\Delta,m} - \log \nu_t^{(j_n)} \right\}.$$

The instantaneous firing rate take the form of a generalized linear model. Each neuron has a baseline rate that is a function of b_n and λ_{max}. Moreover, the rate is influenced by recent spiking activity through the network, $\boldsymbol{A} \odot \boldsymbol{W}$, and through the local normalization, ν.

According to our theory of neural inference, the sparsity pattern of the network should follow an independent Bernoulli model. That is, each edge is present with the same probability, $a_{m\to n} \sim \mathrm{Bern}(\rho)$. Moreover, the weights of network should encode the pairwise log probabilities, $\log \phi(z_i = k, z_j = k')$, and the weights from the R neurons representing $z_i = k$ to the R neurons representing $z_j = k'$ should all be approximately equal. This corresponds to stochastic block structure in the weight matrix. Thus, to reverse engineer the probabilistic model from the observed spike train, we fit a generalized linear model with a spike-and-slab network prior that has an independent Bernoulli model for the adjacency matrix, and a stochastic block model (SBM) for the weight matrix. The SBM has latent variables for each neuron that indicate the cluster assignment, and parameters that specify the average weight between each pair of clusters:

$$\begin{aligned} c_n &\sim \mathrm{Discrete}(\alpha \mathbf{1}), \\ \mu_{c\to c'} &\sim \mathcal{N}(0, \sigma_0^2), \\ w_{m\to n} &\sim \mathcal{N}(\mu_{c_m \to c_n}, \sigma^2). \end{aligned}$$

By performing Bayesian inference in this model, we recover a posterior distribution over biases, $\{b_n\}_{n=1}^N$; weighted adjacency matrices, $\boldsymbol{A}$ and $\boldsymbol{W}$; latent variables, $\{c_n\}_{n=1}^N$; and parameters, $\{\mu_{c\to c'}\}_{c,c'=1}^C$.

The normalization presents a minor complication. According to the theory, this is most likely computed by local inhibitory neurons that estimate the population rate of neurons representing z_j and deliver a common, normalizing input to stabilize the rate at the desired level. This can be roughly approximated with direct, excitatory connections between neurons representing the same variable-value pair, and inhibitory connections between neurons representing competing values of the same variable. In other words, if we focus solely on the activity of neurons representing the variable-value pairs, we should expect additional *functional* connections that encode the mutually exclusive nature of the distinct values of a given variable.

We demonstrate this approach by fitting the hierarchical model to a neural spike train simulated from the population described in Section 9.5. The population performs inference in a simple mixture model with three latent mixture components, and five binary observations. An observation consists of an assignment of the five observations, indicated by the variables $\{x_j\}_{j=1}^5$, and given an observation, the dynamics perform posterior inference of z, the latent variable indicating the underlying mixture component. The population consists of 130 neurons, 10 for each variable-value pair.

Figure 9.5 shows the inferred parameters of the hierarchical model. The posterior mean of the weighted adjacency matrix is shown in Figure 9.5a, with neurons sorted by variable and then by value. The block diagonal structure shows the normalizing connections between neurons representing the same variable, and the region highlighted in yellow shows the inferred connections from neurons representing x_j to neurons representing z. These encode the pairwise log probabilities of the mixture model.

Figure 9.5b shows the inferred posterior probability of two neurons belonging to the same cluster. While the true model has 13 clusters, we allowed our model to use as many as 20 clusters. If the variable-value subpopulations were recovered perfectly, this matrix would be block diagonal. We see that it is nearly so; only a handful of neurons are misclassified and some blocks are split in two.

Finally, Figure 9.5c shows the true and inferred mean weights under the stochastic block model. First, we found the permutation of inferred cluster labels that best matched the true cluster labels. Then we found the linear transformation that best matched the true and inferred weights. The result shows that the true pattern of weights from x_j to z are recovered with high fidelity.

While this procedure for reverse engineering probabilistic models from observed spike trains is

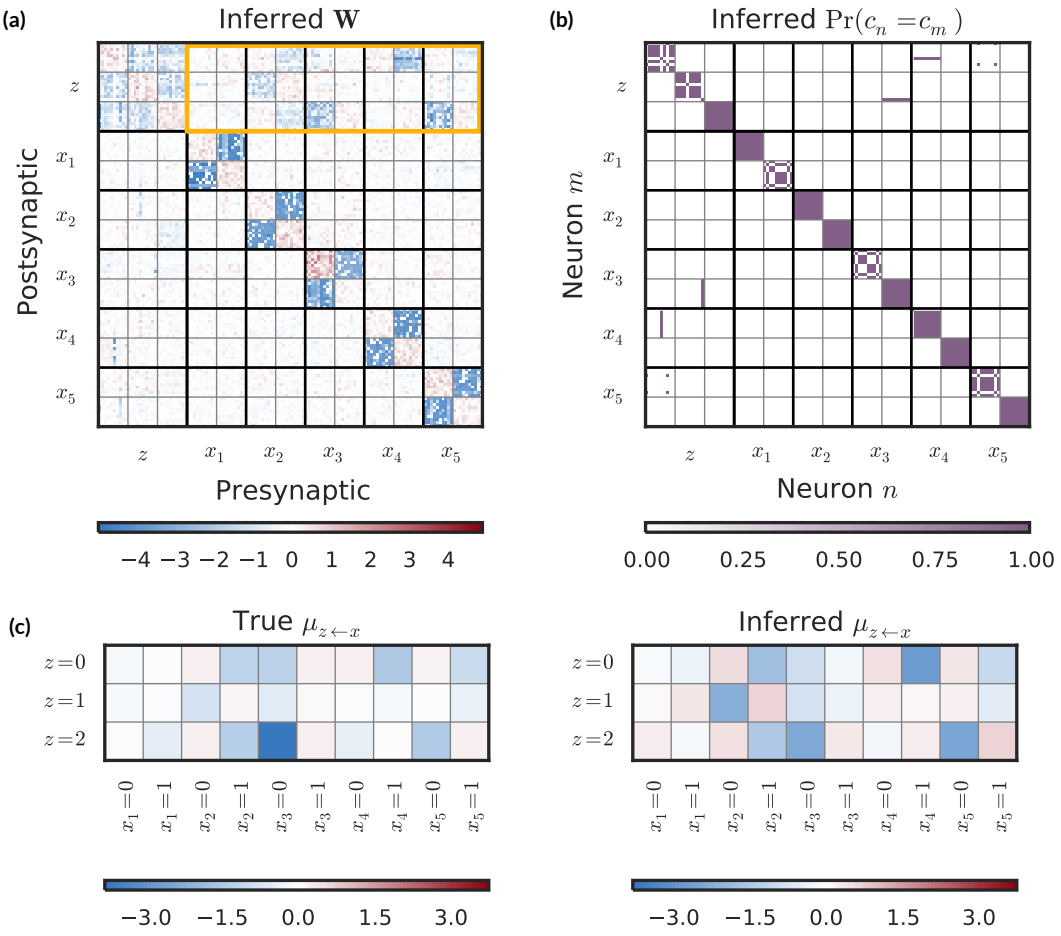

Figure 9.5: The probabilistic model can be reverse engineered from the neural spike train. **(a)** Inferred weighted adjacency matrix for the population of 130 neurons. Thin lines delineate boundaries between subpopulations for each value of z and x_j; bold lines separate populations for each variable. **(b)** Inferred probability that each pair of neurons belongs to the same cluster under a stochastic block model. The block diagonal structure shows that the variable-value subpopulations are clearly recovered. **(c)** True and inferred weights from x_j to z (yellow square in **(a)**). Inferred weights are the mean weights under the stochastic block model. They accurately recover the true weights.

not foolproof, this simple example illustrates that much can be learned by combining top-down theories with bottom-up analysis. To further improve this inference procedure, we should include two sets of latent cluster assignments in our hierarchical model: one set of variables that specifies the variable that a cluster of neurons represents (i.e. j_n in our theory), and another that indicates the value (i.e. k_n). Incorporating the knowledge that different values of the same variable are mutually exclusive, we can build a strong prior distribution over weights given these two variables.

What else can be learned from the results of this approach? In practice, we only observe a fraction of the neurons in a particular region If our recording method samples N neurons out of N_{total}, then given an inferred block size, $\widehat{R}$, we can estimate the true representation size to be roughly, $R \approx \widehat{R} N_{\text{total}}/N$. If variable-value subpopulations are truly disjoint, this provides an estimate of the number of subpopulations the region could encode. Combined with the complexity theoretic bounds developed in Section 9.3, these top-down and bottom-up approaches provide two tacks by which we may converge on a theory of probabilistic inference in neural circuits.

9.7 Future Work

This chapter has illustrated how theoretical models of neural computation may be assessed from a "top-down" perspective by analyzing the complexity and comparing it to biological parameters, and from the "bottom-up" perspective, by incorporating theoretical dynamics into probabilistic models of neural population activity. This is only a first step toward closing the gap between these two perspectives, and it opens many important questions. We enumerate and partially answer a few of them here.

9.7.1 Unsupervised Learning via Synaptic Plasticity

Perhaps the most pressing question is that of learning: how are these subpopulations of neurons and their weighted connections established? While we do not have a concrete answer to this question, we speculate that subpopulations are primarily allocated in a supervised fashion by a process like those of Valiant (1994). Once the neurons have been allocated, their weights may be tuned in an unsupervised manner. We suggest one way in which this unsupervised learning may be related to the process of spike-timing dependent plasticity, based on the work of Nessler et al. (2013).

The parameters of the model, $\boldsymbol{\theta}$, specify the conditional probabilities for pairs of hidden and visible variables. Rather than treating the parameters as given, we now treat them as part of the model.

$$\begin{aligned} p(\boldsymbol{z}, \boldsymbol{\theta}) &= p(\boldsymbol{\theta})\, p(\boldsymbol{z} \mid \boldsymbol{\theta}) \\ &= \frac{p(\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \prod_{j \in \mathcal{G}} \phi(z_j \mid \boldsymbol{\theta}) \prod_{i,j \in \mathcal{G}} \phi(z_i, z_j \mid \boldsymbol{\theta}). \end{aligned}$$

The challenge with learning is that the parameters appear in the normalizing constant, $Z(\boldsymbol{\theta})$, which is typically an intractable summation over variable assignments. For this simple example, we will only consider learning in a subset of models that can formulated as directed graphical models.

Assumption 5. *The following unsupervised learning algorithm assumes that the probabilistic model not only factors into the product of unary and pairwise potentials, but that this factorization corresponds to a directed graphical model in which the variables have at most one "parent" variable. That is, the variables are ordered such that the joint probability is equal to,*

$$p(\boldsymbol{z}, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{j=1}^{J} p(z_j \mid \mathsf{pa}(z_j), \boldsymbol{\theta}),$$

where $\mathsf{pa}(z_j) \in \{\varnothing, z_1, \ldots, z_{j-1}\}$. *Each conditional distribution in this product is properly normalized, which implies that the joint distribution is normalized as well.*

While this is clearly a strict assumption, it allows for some realistic models like mixtures and hidden Markov models. The advantage is that, here, the distribution is normalized such that the parameters appear only in their prior and in the conditional distributions, which depend on at most two variables. This will map nicely onto synaptic plasticity rules.

Since we are assuming the variables are discrete, the parameters $\boldsymbol{\theta}$ specify either the marginal probability of z_j (if $\mathsf{pa}(z_j) = \varnothing$) or the rows of a conditional probability table (if $\mathsf{pa}(z_j) \in \{z_1, \ldots, z_{j-1}\}$). We make this explicit with the following notation,

$$p(z_j \mid \mathsf{pa}(z_j) = \varnothing, \boldsymbol{\theta}) = \mathrm{Discrete}(\boldsymbol{\theta}^{(j)}),$$
$$p(z_j \mid \mathsf{pa}(z_j) = z_{j'} = k, \boldsymbol{\theta}) = \mathrm{Discrete}(\boldsymbol{\theta}^{(j,k)}).$$

In words, if the variable z_j has no parent, it is marginally distributed according to a categorical distribution with parameter $\boldsymbol{\theta}^{(j)}$. If variable z_j has parent $z_{j'}$, then when $z_{j'} = k$, the variable z_j follows a categorical distribution with parameter $\boldsymbol{\theta}^{(j,k)}$.

To incorporate these parameters into the model, we introduce Dirichlet priors over the probability vectors,

$$\boldsymbol{\theta}^{(j)} \sim \mathrm{Dir}(\alpha \mathbf{1}), \qquad \boldsymbol{\theta}^{(j,k)} \sim \mathrm{Dir}(\alpha \mathbf{1}).$$

Learning in a Bayesian framework corresponds to performing posterior inference over the parameters. Thus, we introduce a variational factor for $\boldsymbol{\theta}$ as well,

$$q(\boldsymbol{\theta}) = \prod_{j:\mathsf{pa}(z_j)=\varnothing} q(\boldsymbol{\theta}^{(j)}) \prod_{j:\mathsf{pa}(z_j)\neq\varnothing} \prod_{k} q(\boldsymbol{\theta}^{(j,k)})$$

Consider the variational factor for $\boldsymbol{\theta}^{(j,k)}$. Omitting the details, we can show that this factor takes the form of a Dirichlet distribution,

$$\begin{aligned} q_t(\boldsymbol{\theta}^{(j,k)}) &= \mathrm{Dir}(\boldsymbol{\theta}^{(j,k)} \mid \boldsymbol{\alpha}_t^{(j,k)}), \\ \boldsymbol{\alpha}_t^{(j,k)} &= \alpha + \widehat{\boldsymbol{\pi}}_{t-1}^{(j)} \cdot \widehat{\pi}_{t-1,k}^{(\mathsf{pa}(z_j))}. \end{aligned}$$

The updates of $q(z_j)$ must consider expectations with respect to this Dirichlet factor:

$$\log q(z_j) \simeq \mathbb{E}_{q(\boldsymbol{z}_{\neg j})} \mathbb{E}_{q(\boldsymbol{\theta})} \left[p(\boldsymbol{z}, \boldsymbol{\theta}) \right].$$

This expectation is given by,

$$\begin{aligned} \mathbb{E}_{q_t(\boldsymbol{\theta})} \left[\log p(z_j = k \mid \mathsf{pa}(z_j) = k', \boldsymbol{\theta}) \right] &= \mathbb{E}_{q_t(\boldsymbol{\theta})} \left[\log \theta_k^{(j,k')} \right] \\ &= \psi\big(\alpha_{t,k}^{(j,k')}\big) - \psi\big(\sum_{i=1}^{K} \alpha_{t,i}^{(j,k')}\big) \\ &= \psi\big(\alpha_{t,k}^{(j,k')}\big) - \psi\big(K\alpha + \widehat{\pi}_{t-1,k'}^{(\mathsf{pa}(z_j))}\big). \end{aligned} \tag{9.6}$$

How could this be implemented biologically? First, we assume that learning occurs on a timescale of T_L time steps, which is relatively slow compared to the time scales of inference and behavior. That is, $T_I < T_L$. This allows the learning algorithm to generalize from many input rather than overfitting to a single example.

We want the synaptic weights to equal the expected log parameter value, as in (9.6). In theory, the weights should be identical for all synapses between neurons representing $(z_j = k)$ and neurons representing $(z_{j'} = k')$. It is unreasonable to assume this in practice, since these synapses exist between different neurons and are updated independently. However, we can specify a simple learning rule that would give rise to the same weights in expectation.

Assume that each synapse has a two latent state variables, $\alpha_{t,m\to n}$, and $\beta_{t,m\to n}$. These will enable us to compute the expectation with respect to the variational parameter. We propose the following learning rule for the first state,

$$\begin{aligned} \alpha_{t,m\to n} &= \alpha + (\lambda_{\mathsf{max}}^2 T_L)^{-1} \sum_{\Delta=1}^{T_L} s_{t-\Delta,n} \cdot s_{t-\Delta,m} \\ &\approx \alpha + \widehat{\pi}_{t-1,k_n}^{(j_n)} \cdot \widehat{\pi}_{t-1,k_m}^{(j_m)}. \end{aligned}$$

Suppose neuron n represents the child variable and neuron m represents the parent variable. Then,[†]

$$\begin{aligned}\beta_{t,m\to n} &= K\alpha + (\lambda_{\mathsf{max}} T_L)^{-1} \sum_{\Delta=1}^{T_L} s_{t-\Delta,m} \\ &\approx K\alpha + \widehat{\pi}_{t-1,k_m}^{(z_{j_m})}.\end{aligned}$$

The synaptic weight is then a deterministic function of these two state variables,

$$w_{t,m\to n} = \psi(\alpha_{t,m\to n}) - \psi(\beta_{t,m\to n}).$$

This state-based learning rule is Hebbian in that correlated spiking activity leads to increases in $\alpha_{t,m\to n}$, which in turn lead to larger weights (since the digamma function is increasing on the nonnegative reals). This is counteracted by the accrual of $\beta_{t,m\to n}$, which counts pre- or post-synaptic spikes, depending on whether the post-synaptic neuron represents the child or parent variable, respectively. If this value is large relative to $\alpha_{t,m\to n}$, the spike correlation is low relative to the background rate, which implies a low probability and a strongly negative weight.

Moreover, this learning rule is nonlinear. While the state variables are linear functions of pre- and post-synaptic spike counts, their effect on the weight is highly nonlinear due to the digamma functions. We could instead write this learning rule as a nonlinear dynamical system on the weights alone since the digamma function is also invertible on this range. Using the tools developed in Chapter 6, this dynamic learning process could potentially be incorporated in a probabilistic model for neural activity as well. We leave this for future work.

9.7.2 Representing Continuous Random Variables

While this chapter has focused on representing discrete random variables, many of the quantities we need to infer and reason about are continuous in nature. Suppose that we wish to represent a random variable $z \in \mathbb{R}^D$. Rather than representing the parameters of a standard distribution, like the mean and variance of a Gaussian, the brain may use a nonparametric representation like a kernel

[†]Here, the synaptic state variable counts spikes on the pre-synaptic neuron. If the parent-child order was flipped, the synapse would have to count post-synaptic spikes instead. This asymmetry is admittedly somewhat unsatisfying.

density estimate for the variational factors (Anderson and Essen, 1994; Barber et al., 2003). Suppose,

$$q_t(z_j) \propto \sum_{k=1}^{K} \eta_{t,k}^{(j)} \, \zeta(z_j; \mu_k),$$

where $\{\eta_{t,k}^{(j)}\}_{k=1}^{K}$ is a set of nonnegative weights that sum to one, $\zeta(z; \mu)$ is a nonnegative "kernel function" that integrates to one and has mean μ, and $\{\mu_k\}$ is the set of means at which these kernels are located. This defines a proper density function because $q_t(z_j)$ is nonnegative and integrates to one,

$$\int q_t(z_j) \, \mathrm{d}z_j = \sum_{k=1}^{K} \int \eta_{t,k}^{(j)} \, \zeta(z; \mu_k) \, \mathrm{d}z_j = \sum_{k=1}^{K} \eta_{t,k}^{(j)} = 1.$$

To implement this with a distributed population of neurons, let

$$\eta_{t,k}^{(j)} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[j_n = j, k_n = k] \, s_{t,n}}{\sum_{n=1}^{N} \mathbb{I}[j_n = j] s_{t,n}}.$$

The mean field variational inference algorithm is no longer as simple as in Section 9.4, but the key quantities of the variational lower bound, namely the entropy of the variational factor and the expected log probability, are still tractable. Using an approach similar to that of Gershman et al. (2012a), we have,

$$\begin{aligned}
\mathbb{E}_{q_t(\boldsymbol{z})} \left[\log p(\boldsymbol{z} \mid \boldsymbol{\theta}) \right] &= \mathbb{E}_{q_t(\boldsymbol{z})} \left[\sum_{j \in \mathcal{G}} \log \phi(z_j \mid \boldsymbol{\theta}) + \sum_{i,j \in \mathcal{G}} \log \phi(z_i, z_j \mid \boldsymbol{\theta}) \right] \\
&= \sum_{j \in \mathcal{G}} \sum_{k=1}^{K} \eta_{t,k}^{(j)} \int \log \phi(z_j \mid \boldsymbol{\theta}) \zeta(z_j; \mu_k) \, \mathrm{d}z_j \\
&\quad + \sum_{i,j \in \mathcal{G}} \sum_{k=1}^{K} \sum_{k'=1}^{K} \eta_{t,k'}^{(i)} \, \eta_{t,k}^{(j)} \iint \log \phi(z_i, z_j \mid \boldsymbol{\theta}) \, \zeta(z_i; \mu_k) \, \zeta(z_j; \mu_{k'}) \, \mathrm{d}z_i \, \mathrm{d}z_j \\
&= \sum_{j \in \mathcal{G}} \sum_{k=1}^{K} \eta_{t,k}^{(j)} \, \widetilde{b}_k^{(j)} + \sum_{i,j \in \mathcal{G}} \sum_{k'=1}^{K} \sum_{k=1}^{K} \eta_{t,k'}^{(i)} \, \eta_{t,k}^{(j)} \, \widetilde{w}_{k',k}^{(i,j)}
\end{aligned}$$

The second term of the variational lower bound is the entropy of the variational distribution,

$$\begin{aligned}\mathcal{H}[q_t(z_j)] &= -\int q_t(z_j) \log q_t(z_j) \, \mathrm{d}z_j \\ &= -\int q_t(z_j) \log \sum_{k=1}^{K} \eta_{t,k}^{(j)} \, \zeta(z_j; \mu_k) \, \mathrm{d}z_j.\end{aligned}$$

We can lower bound this with Jensen's inequality to get,

$$\begin{aligned}\mathcal{H}[q_t(z_j)] &\geq \sum_{k=1}^{K} \log \int q_t(z_j) \, \eta_{t,k}^{(j)} \, \zeta(z_j; \mu_k) \, \mathrm{d}z_j \\ &= \sum_{k=1}^{K} \log \left(\eta_{t,k}^{(j)} \sum_{k'=1}^{K} \eta_{t,k'}^{(j)} \int \zeta(z_j; \mu_{k'}) \, \zeta(z_j; \mu_k) \, \mathrm{d}z_j \right) \\ &= \sum_{k=1}^{K} \log \left(\eta_{t,k}^{(j)} \sum_{k'=1}^{K} \eta_{t,k'}^{(j)} \, \zeta_{k,k'}^{*} \right)\end{aligned}$$

where we have defined $\zeta_{k,k'}^{*} = \zeta_{k',k}^{*} = \int \zeta(z_j; \mu_{k'}) \, \zeta(z_j; \mu_k) \, \mathrm{d}z_j$ as the convolution of a pair of kernel functions.

Now, to perform inference, we can perform gradient ascent directly on the *evidence lower bound* (ELBO) on the log marginal likelihood, which is just the sum of these two terms. That is, we drive firing rates such that,

$$\eta_{t,k}^{(j)} = \eta_{t-1,k}^{(j)} + \alpha \nabla_{\boldsymbol{\eta}} \left(\mathbb{E}_{q_{t-1}(\boldsymbol{z})} \left[\log p(\boldsymbol{z} \mid \boldsymbol{\theta}) \right] + \mathcal{H}[q_{t-1}(z_j)] \right).$$

The gradient of the ELBO has two components, the first from the expected log probability and the second from the entropy. The first is quite intuitive,

$$\frac{\partial}{\partial \eta_{t,k}^{(j)}} \mathbb{E}_{q_t(\boldsymbol{z})} \left[\log p(\boldsymbol{z} \mid \boldsymbol{\theta}) \right] = \widetilde{b}_k^{(j)} + \sum_{i \in \mathsf{ne}(j)} \sum_{k'=1}^{K} \eta_{t,k'}^{(i)} \, \widetilde{w}_{k',k}^{(i,j)}.$$

As in the discrete random variable case, the firing rates of neurons representing the pair (j, k), are driven by a bias and a weighted sum of activity from connected neurons. Now, however, the bias and the weights reflect integrations with respect to the basis functions.

The second term comes from the lower bound on the entropy,

$$\begin{aligned}\frac{\partial}{\partial \eta_{t,k}^{(j)}} \mathcal{H}[q_t(z_j)] &= \frac{1}{\eta_{t,k}^{(j)}} + \sum_{k' \neq k} \frac{\partial}{\partial \eta_{t,k}^{(j)}} \log \left(\eta_{t,k}^{(j)} \zeta_{k',k}^* + \sum_{k'' \neq k} \eta_{t,k''}^{(j)} \zeta_{k',k''}^* \right) \\ &= \frac{1}{\eta_{t,k}^{(j)}} + \sum_{k' \neq k} \zeta_{k',k}^* \left(\textstyle\sum_{k''=1}^{K} \eta_{t,k''}^{(j)} \zeta_{k',k''}^* \right)^{-1}. \end{aligned} \tag{9.7}$$

While less intuitive, this term effectively provides a damping signal that prevents one kernel from dominating the rest. As the rates approach one, the first term in (9.7) diminishes. We leave more detailed studies of the biological plausibility of this approach to future work.

9.7.3 Alternative Representations of Probability

The introduction enumerated a host of potential neural representations of probability and corresponding inference algorithms. Eventually, these combinations of representation and algorithm lead to some prediction of the dynamics of neural spiking. Often, these dynamics follow standard forms, like the linear-nonlinear cascade of the GLM. If this is the case, then we should be able to derive probabilistic models for neural data that incorporate the hypothesized dynamics of Bayesian inference.

One popular theory of representation is the *probabilistic population code* (PPC) (Ma et al., 2006). According to this theory, the Poisson-like variability of neurons leads to a likelihood of a random variable for any particular spike train, $p(\boldsymbol{s} \mid z_j)$. Combined with a prior, this yields a posterior distribution, $p(z_j \mid \boldsymbol{s})$. In their theory, the encoded distribution is exactly this posterior, $\widehat{p}(z_j) = p(z_j \mid \boldsymbol{s})$.

This leads to two levels of randomness. While the neurons may be driven with firing rates that, in expectation, encode the distribution $\bar{p}(z_j)$, the randomness in $\boldsymbol{s}$ implies a distribution over encoded distributions, $p(\widehat{p}(z_j))$. This doubly stochastic nature has been explored by Zemel et al. (1998), Sahani and Dayan (2003), and others. Ma et al. (2006) skirt this issue by assuming that as the number of neurons grows, this distribution over distributions collapses to its mode, $p(\widehat{p}(z_j)) = \delta_{p^*(z_j)}$, which is presumed to be approximately equal to the desired distribution. That is, $p^*(z_j) \approx \bar{p}(z_j)$. In their example, a Gaussian distribution is encoded by a population of neurons with radial basis function tuning curves. The mean is encoded by the relative firing rate of the activity, and the precision is encoded by the absolute firing rate, or gain, of the population.

Their main contribution is a demonstration of how inference in some simple probabilistic mod-

els, like a naïve Bayes model for cue combination, can be performed with simple linear functions on PPCs. For example, in simple naïve Bayes models, downstream neurons only need to sum population activity to combine evidence and compute an updated posterior. However, other probabilistic computations, like marginalization and variational inference, do require nonlinear operations (Beck et al., 2011; 2012).

If neural populations are performing inference with PPCs using linear operations, then the neural spike trains recorded from these populations should follow linear Hawkes process dynamics. Thus, the tools of Chapters 2 and 3 could provide a mechanism for inferring the connection weights. Given these connection weights and an estimate of the tuning curves, it should be possible to reverse engineer the underlying probabilistic model. This provides one more avenue toward connecting theory and experiment.

9.8 Conclusion

This chapter has taken a novel look at the problem of connecting the theory of neural computation to experimental recording. While the traditional approach of making specific, testable predictions of a theory remains invaluable, this is a process we would like to automate as much as possible. With the advent of large-scale recording technologies, the bottleneck in the scientific process moves from collecting evidence to designing experiments and revising theories. Here, we have suggested that the scientific loop of theorizing, experimenting, and revising may be closed by formulating our theories in the language of prior distributions in a Bayesian probabilistic model of neural data. With such a model, we could hypothetically measure the marginal likelihood of a theory given the data, suggest experiments to refine our estimate of theoretical values, and revise our theories in an automated fashion. This chapter has not nearly closed this loop, but it has provided a framework for thinking about a future in which theoretical, computational, and systems neuroscience are tightly tethered.

10

Conclusion

Recent advances in neural recording technologies have stirred great excitement. It seems that a more complete understanding of neural computation is within our grasp. Armed with these powerful tools, we can peer into the brain and observe the activity of most, if not all, of the neurons in a circuit. What a major advance over the handfuls of neurons we were limited to only a few years ago! All we must do is extract the underlying patterns and principles from these large scale recordings.

While these advances present unprecedented opportunities, the task of translating data into understanding is far from trivial. The human genome has been known for a decade now (Consortium, 2004; Gregory et al., 2006), yet much of its structure remains enigmatic. The connectome of the nematode *C. Elegans* has been known for three decades (White et al., 1986), yet our understanding of this simple organism with fewer than 400 neurons is still incomplete. At the heart of these endeavors is the search for meaningful abstractions and structured representations given complex and noisy data. Our success in reverse engineering neural computation relies critically on our ability to discover such structure. This thesis has developed a number of methods for instantiating structural hypotheses in the form of probabilistic models and turning the crank of Bayesian inference in order to reason about them.

I believe the path forward lies in the iterative refinement of theories guided by both top-down considerations of algorithmic goals and complexity-theoretic constraints, and bottom-up, data-driven analyses of neural data. The Bayesian methods presented in this thesis are designed to acceler-

ate this process, providing "data microscopes" that allow us to visualize complex, high-dimensional data in new and interpretable ways. I will briefly discuss two directions in which these methods should continue to be developed.

10.1 Toward Programmatic Models of Neural Computation

As we have shown here, hierarchical probabilistic models provide an intuitive language for capturing different types of abstraction, allowing us to formalize generative processes of how data comes to be. However, as these models grow in scope and scale, the language of probabilistic models becomes cumbersome. At the same time, as our models grow in complexity, they look more and more like *probabilistic programs* (Goodman et al., 2008).

The probabilistic models developed in this thesis can all be written in this way. For example, the hidden Markov models of Chapter 7 are equivalent to the following program:

Require: $\boldsymbol{\pi}^{(0)}, \boldsymbol{P}, \boldsymbol{\Lambda}$
  for $t = 1, \ldots, T$ **do**
    if $t = 1$ **then**
      $z_t \sim \text{Discrete}(\boldsymbol{\pi}^{(0)})$
    else
      $z_t \sim \text{Discrete}(\boldsymbol{\pi}^{(z_{t-1})})$
    end if
    for $n = 1, \ldots, N$ **do**
      $s_{t,n} \sim \text{Poisson}(\lambda_{z_t,n})$
    end for
  end for

Program 10.1: Programmatic representation of a hidden Markov model.

This representation is equivalent to the probabilistic model (it implies the same distribution over $\boldsymbol{z}$ and $\boldsymbol{S}$), but this description combines stochastic operations, like sampling, with basic control flow, like **if** statements and **for** loops. This powerful combination not only enables rapid formulation of models for neural data, it also forms the basis for a "probabilistic language of thought," an idea that is taking hold in cognitive science (Goodman et al., 2014). As we seek to bridge the gap between cognitive algorithms and neural implementations, it will help if we are speaking the same language.

Of course, the "no free lunch" theorem applies here as well. While probabilistic programming languages make it easy to specify complex generative processes, they make it just as easy to specify

models for which Bayesian inference is completely intractable. While much progress has been made in general purpose inference algorithms (Goodman et al., 2008; Ranganath et al., 2014; Mansinghka et al., 2014; Wood et al., 2015; Kucukelbir et al., 2015), these "black box" inference algorithms are, by design, not capitalizing on model-specific structure that the rather bespoke inference algorithms of this thesis have leveraged. This will certainly change as probabilistic program "compilers" become more adept at recognizing model structure, but this is currently a major challenge.

10.2 Toward Joint Models of Neural Activity, Behavior, and Environment

This thesis has focused solely on modeling the dynamics and structure of neural spike trains, however, this data is often collected from organisms as they perform natural behaviors in complex behaviors. For example, massive recordings are now being collected from animals in decision making (e.g. Briggman et al., 2005), freely behaving (e.g. Prevedel et al., 2014), and evoked response (e.g. Portugues et al., 2014) tasks. This type of data provides a tremendous opportunity to study the relationship between neural activity, natural behavior, and environment. But first, we must formulate and fit a model that captures both the complex dynamics of neural activity, the rich repertoire of behavior, and the environmental state. The models and inference algorithms designed in this thesis capture core notions of state and dynamics that can be extended, in an intuitive way, to these types of recordings.

For example, large-scale recordings have revealed that ensembles of neurons reliably participate together during natural or trained behavior, suggesting that task-related neural activity might be lower-dimensional than the number of recorded neurons, and that these neurons might evolve through different states over time in an environmentally dependent manner. The dynamical system models developed in this thesis can naturally instantiate these hypotheses. Consider a model in which neural spike trains, $\boldsymbol{s}_t$, behavioral time series, $\boldsymbol{b}_t$, and environmental stimuli, $\boldsymbol{e}_t$, are simultaneously measured. A simple hypothesis is that the neural spike trains and the behavior are conditionally independent given underlying states, $\boldsymbol{x}_t$ and z_t, and that the evolution of these states depends on the environment, $\boldsymbol{e}_t$. Such a model enables us to identify the low dimensional states of neural activity and overt behavior, as well as their dynamics. Moreover, it enables us to predict one given the others. With relatively minor adjustments, the inference algorithms developed previously can be extended to handle these multimodal datasets.

Ultimately, the goal of computational and systems neuroscience is to understand this interplay between environment, neural activity, and behavior. As with all scientific endeavors, our success

will be measured in our ability to articulate theories of neural computation that explain, in simpler terms, the complex nature of these multifaceted systems. With the advent of recording technologies capable of probing neural circuits at unprecedented scale and advances in machine learning providing the computational and statistical tools for making sense of complex data, it seems the stage is set for major breakthroughs in our understanding of nature's most sophisticated computer: the human brain.

A

Common Distributions

This thesis makes use of a number of common distributions. The notation $z \sim \mathrm{P}(\theta)$ means that the random variable z is sampled from (or distributed according to) the distribution P, which is parameterized by θ. When we write $\mathrm{P}(z \mid \theta)$ we refer to the density (assuming it exists) of P evaluated at z. Here, we provide a summary of common distributions and their parametric densities or mass functions.

Bernoulli

For a binary random variable $x \in \{0, 1\}$ with $\rho \in [0, 1]$,

$$\mathrm{Bern}(x \mid \rho) = \rho^x (1 - \rho)^{1-x}.$$

Beta

For a continuous random variable $\rho \in [0, 1]$ with $a > 0$ and $b > 0$,

$$\mathrm{Beta}(\rho \mid a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \rho^{a-1} (1 - \rho)^{b-1}.$$

The beta distribution is a conjugate prior for the Bernoulli, binomial, and negative binomial distributions.

Binomial

For an integer-valued random variable $x \in \{1, \ldots, N\}$ with $N \in \mathbb{N}$ and $\rho \in [0, 1]$,

$$\mathrm{Bin}(x \mid N, \rho) = \binom{N}{x} \rho^x (1 - \rho)^{N-x}.$$

Dirichlet

For a probability vector $\boldsymbol{\pi} \in [0, 1]^K$ such that $\pi_k \geq 0$ and $\sum_k \pi_k = 1$, and parameter $\boldsymbol{\alpha} \in \mathbb{R}_+^K$,

$$\mathrm{Dir}(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) = \frac{\Gamma\big(\sum_{k=1}^K \alpha_k\big)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \pi_k^{\alpha_k - 1}.$$

The Dirichlet distribution is a conjugate prior to the discrete and multinomial distributions.

Discrete

For a discrete random variable $x \in \{1, \ldots, K\}$ with K distinct outcomes, and a probability vector $\boldsymbol{\pi} \in [0, 1]^K$ that is nonnegative and sums to one,

$$\mathrm{Discrete}(x \mid \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{\mathbb{I}[x=k]}.$$

Gamma

For a nonnegative random variable $\lambda \in \mathbb{R}_+$ with shape parameter $a > 0$ and rate parameter $b > 0$,

$$\mathrm{Gamma}(\lambda \mid a, b) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda}.$$

The gamma distribution is the conjugate prior to the Poisson distribution, as well as to the rate parameter of the gamma distribution. The gamma distribution may also be parameterized in terms of a scale parameter, $\theta = b^{-1}$, but we do not use that parameterization in this thesis.

Gaussian

For a random variable $\boldsymbol{x} \in \mathbb{R}^D$ with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and positive semidefinite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$,

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-D/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right\}.$$

Multinomial

For a vector of discrete counts $\boldsymbol{x} \in \mathbb{N}^K$ with $\sum_k x_k = N$ and a probability vector $\boldsymbol{\pi} \in [0, 1]^K$,

$$\mathrm{Mult}(\boldsymbol{x} \mid N, \boldsymbol{\pi}) = \binom{N}{x_1, x_2, \ldots x_K} \prod_{k=1}^{K} \pi_k^{x_k},$$

where

$$\binom{N}{x_1, x_2, \ldots x_K} = \frac{N!}{x_1! \ldots x_K!}.$$

Negative Binomial

For an integer-valued random variable $x \in \mathbb{N}$ with shape parameters $\nu \in \mathbb{R}_+$ and probability $\rho \in [0, 1]$,

$$\mathrm{NB}(x \mid \nu, \rho) = \binom{x + \nu - 1}{x} \rho^x (1 - \rho)^\nu.$$

Poisson

For an integer random variable $x \in \mathbb{N}$ and a nonnegative rate parameters $\lambda \in \mathbb{R}_+$,

$$\mathrm{Poisson}(x \mid \lambda) = \frac{1}{x!} \lambda^x e^{-\lambda}.$$

Uniform

For a continuous random variable $x \in \mathbb{R}$,

$$\text{Unif}(x \mid a, b) = \begin{cases} \frac{1}{b-a} & \text{if } a < x < b, \\ 0 & \text{o.w.}. \end{cases}$$

References

Yashar Ahmadian, Jonathan W Pillow, and Liam Paninski. Efficient Markov chain Monte Carlo methods for decoding neural spike trains. *Neural Computation*, 23(1):46–96, 2011.

Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, 10 (5):413–420, 2013.

Laurence Aitchison and Peter E Latham. Synaptic sampling: A connection between PSP variability and uncertainty explains neurophysiological observations. *arXiv preprint arXiv:1505.04544*, 2015.

Laurence Aitchison and Máté Lengyel. The Hamiltonian brain. *arXiv preprint arXiv:1407.0973*, 2014.

David J Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

Charles H Anderson and David C Van Essen. Neurobiological computational systems. *Computational Intelligence Imitating Life*, pages 1–11, 1994.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3): 269–342, 2010.

Michael J Barber, John W Clark, and Charles H Anderson. Neural representation of probabilistic information. *Neural Computation*, 15(8):1843–64, August 2003.

Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen. The infinite hidden Markov model. *Advances in Neural Information Processing Systems 14*, pages 577–585, 2002.

Jeffrey M Beck and Alexandre Pouget. Exact inferences in a neural implementation of a hidden Markov model. *Neural Computation*, 19(5):1344–1361, 2007.

Jeffrey M Beck, Peter E Latham, and Alexandre Pouget. Marginalization in neural circuits with divisive normalization. *The Journal of Neuroscience*, 31(43):15310–15319, 2011.

Jeffrey M Beck, Katherine A Heller, and Alexandre Pouget. Complex inference in neural circuits with probabilistic population codes and topic models. *Advances in Neural Information Processing Systems*, pages 3059–3067, 2012.

Yoshua Bengio and Paolo Frasconi. An input output HMM architecture. *Advances in Neural Information Processing Systems*, pages 427–434, 1995.

Pietro Berkes, Gergo Orbán, Máté Lengyel, and József Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–7, January 2011.

Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99): 20140672, 2014.

Philippe Biane, Jim Pitman, and Marc Yor. Probability laws related to the Jacobi theta and Riemann zeta functions, and Brownian excursions. *Bulletin of the American Mathematical Society*, 38 (4):435–465, 2001.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

David M Blei. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1:203–232, 2014.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

Carolyn R Block and Richard Block. *Street gang crime in Chicago*. US Department of Justice, Office of Justice Programs, National Institute of Justice, 1993.

Carolyn R Block, Richard Block, and Illinois Criminal Justice Information Authority. Homicides in Chicago, 1965-1995. ICPSR06399-v5. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], July 2005.

Charles Blundell, Katherine A Heller, and Jeffrey M Beck. Modelling reciprocating relationships with Hawkes processes. *Advances in Neural Information Processing Systems*, pages 2600–2608, 2012.

George EP Box. Sampling and Bayes' inference in scientific modelling and robustness. *Journal of the Royal Statistical Society. Series A (General)*, pages 383–430, 1980.

David H Brainard and William T Freeman. Bayesian color constancy. *Journal of the Optical Society of America A*, 14(7):1393–1411, 1997.

Kevin L Briggman, Henry DI Abarbanel, and William B Kristan. Optical imaging of neuronal populations during decision-making. *Science*, 307(5711):896–901, 2005.

David R. Brillinger. Maximum likelihood analysis of spike trains of interacting nerve cells. *Biological Cybernetics*, 59(3):189–200, August 1988.

David R Brillinger, Hugh L Bryant Jr, and Jose P Segundo. Identification of synaptic interactions. *Biological Cybernetics*, 22(4):213–228, 1976.

Michael Bryant and Erik B Sudderth. Truly nonparametric online variational inference for hierarchical Dirichlet processes. *Advances in Neural Information Processing Systems 25*, pages 2699–2707, 2012.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7(11):e1002211, November 2011.

Lars Buesing, Jakob H. Macke, and Maneesh Sahani. Learning stable, regularised latent models of neural population dynamics. *Network: Computation in Neural Systems*, 23:24–47, 2012a.

Lars Buesing, Jakob H Macke, and Maneesh Sahani. Spectral learning of linear dynamics from generalised-linear observations with application to neural population data. *Advances in Neural Information Processing Systems*, pages 1682–1690, 2012b.

Lars Buesing, Timothy A Machado, John P Cunningham, and Liam Paninski. Clustered factor analysis of multineuronal spike data. *Advances in Neural Information Processing Systems*, pages 3500–3508, 2014.

Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

Santiago Ramón Cajal. *Textura del Sistema Nervioso del Hombre y los Vertebrados*, volume 1. Imprenta y Librería de Nicolás Moya, Madrid, Spain, 1899.

Natalia Caporale and Yang Dan. Spike timing-dependent plasticity: a Hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46, 2008.

Nick Chater and Christopher D Manning. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10(7):335–344, 2006.

Zhe Chen, Fabian Kloosterman, Emery N Brown, and Matthew A Wilson. Uncovering spatial topology represented by rat hippocampal population neuronal codes. *Journal of Computational Neuroscience*, 33(2):227–255, 2012.

Zhe Chen, Stephen N Gomperts, Jun Yamamoto, and Matthew A Wilson. Neural representation of spatial topology in the rodent hippocampus. *Neural Computation*, 26(1):1–39, 2014.

Sharat Chikkerur, Thomas Serre, Cheston Tan, and Tomaso Poggio. What and where: A Bayesian inference theory of attention. *Vision Research*, 50(22):2233–2247, 2010.

Yoon Sik Cho, Aram Galstyan, Jeff Brantingham, and George Tita. Latent point process models for spatial-temporal networks. *arXiv:1302.2671*, 2013.

International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, 2004.

Aaron C Courville, Nathaniel D Daw, and David S Touretzky. Bayesian theories of conditioning in a changing world. *Trends in Cognitive Sciences*, 10(7):294–300, 2006.

Ronald L Cowan and Charles J Wilson. Spontaneous firing patterns and axonal projections of single corticostriatal neurons in the rat medial agranular cortex. *Journal of Neurophysiology*, 71(1): 17–32, 1994.

W Maxwell Cowan, Thomas C Südhof, and Charles F Stevens. *Synapses*. Johns Hopkins University Press, 2003.

Mary Kathryn Cowles and Bradley P Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91:883–904, 1996.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014.

Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: Volume I: Elementary Theory and Methods*. Springer Science & Business Media, 2 edition, 2003.

Peter Dayan and Larry F Abbott. *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT Press, 2001.

Peter Dayan and Joshua A Solomon. Selective Bayes: Attentional load and crowding. *Vision Research*, 50(22):2248–2260, 2010.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

Sophie Deneve. Bayesian spiking neurons I: inference. *Neural Computation*, 20(1):91–117, January 2008.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, USA, 1986.

Christopher DuBois, Carter Butts, and Padhraic Smyth. Stochastic block modeling of relational event dynamics. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 238–246, 2013.

Seif Eldawlatly, Yang Zhou, Rong Jin, and Karim G Oweiss. On the use of dynamic Bayesian networks in reconstructing functional neuronal networks from spike train ensembles. *Neural Computation*, 22(1):158–189, 2010.

Marc O Ernst and Martin S Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, 2002.

Sean Escola, Alfredo Fontanini, Don Katz, and Liam Paninski. Hidden Markov models for the stimulus-response relationships of multistate neural systems. *Neural Computation*, 23(5):1071–1132, 2011.

Warren John Ewens. Population genetics theory—the past and the future. In S. Lessard, editor, *Mathematical and Statistical Developments of Evolutionary Theory*, pages 177–227. Springer, 1990.

Daniel E Feldman. The spike-timing dependence of plasticity. *Neuron*, 75(4):556–71, August 2012.

Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991.

Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.

Christopher R Fetsch, Amanda H Turner, Gregory C DeAngelis, and Dora E Angelaki. Dynamic reweighting of visual and vestibular cues during self-motion perception. *The Journal of Neuroscience*, 29(49):15601–15612, 2009.

Christopher R Fetsch, Alexandre Pouget, Gregory C DeAngelis, and Dora E Angelaki. Neural correlates of reliability-based cue weighting during multisensory integration. *Nature Neuroscience*, 15(1):146–154, 2012.

József Fiser, Pietro Berkes, Gergő Orbán, and Máté Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, 14(3):119–130, 2010.

Alyson K Fletcher, Sundeep Rangan, Lav R Varshney, and Aniruddha Bhargava. Neural reconstruction with approximate message passing (neuramp). *Advances in Neural Information Processing Systems*, pages 2555–2563, 2011.

Emily B Fox. *Bayesian nonparametric learning of complex dynamical phenomena*. PhD thesis, Massachusetts Institute of Technology, 2009.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. An HDP-HMM for systems with state persistence. *Proceedings of the International Conference on Machine Learning*, pages 312–319, 2008.

Jeremy Freeman, Greg D Field, Peter H Li, Martin Greschner, Deborah E Gunning, Keith Mathieson, Alexander Sher, Alan M Litke, Liam Paninski, Eero P Simoncelli, et al. Mapping nonlinear receptive field structure in primate retina at single cone resolution. *eLife*, 4:e05241, 2015.

Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews. Neuroscience*, 11 (2):127–38, February 2010.

Karl J Friston. Functional and effective connectivity in neuroimaging: a synthesis. *Human Brain Mapping*, 2(1-2):56–78, 1994.

Deep Ganguli and Eero P Simoncelli. Implicit encoding of prior probabilities in optimal neural populations. *Advances in Neural Information Processing Systems*, pages 6–9, 2010.

Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, 2015.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. CRC press, 3rd edition, 2013.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

Felipe Gerhard, Tilman Kispersky, Gabrielle J Gutierrez, Eve Marder, Mark Kramer, and Uri Eden. Successful reconstruction of a physiological circuit with known connectivity from spiking activity alone. *PLoS Computational Biology*, 9(7):e1003138, 2013.

Samuel J Gershman, Matthew D Hoffman, and David M Blei. Nonparametric variational inference. *Proceedings of the International Conference on Machine Learning*, pages 663–670, 2012a.

Samuel J Gershman, Edward Vul, and Joshua B Tenenbaum. Multistability and perceptual inference. *Neural Computation*, 24(1):1–24, 2012b.

Sebastian Gerwinn, Jakob Macke, Matthias Seeger, and Matthias Bethge. Bayesian inference for spiking neuron models with a sparsity prior. *Advances in Neural Information Processing Systems*, pages 529–536, 2008.

Charles J Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, pages 473–483, 1992.

Walter R Gilks. *Markov Chain Monte Carlo*. Wiley Online Library, 2005.

Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1019–1028, 2010.

Noah Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church: a language for generative models. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 220––229, 2008.

Noah D Goodman, Joshua B Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. Technical report, Center for Brains, Minds and Machines (CBMM), 2014.

Agnieszka Grabska-Barwinska, Jeff Beck, Alexandre Pouget, and Peter Latham. Demixing odors-fast inference in olfaction. *Advances in Neural Information Processing Systems*, pages 1968–1976, 2013.

SG Gregory, KF Barlow, KE McLay, R Kaul, D Swarbreck, A Dunham, CE Scott, KL Howe, K Woodfine, CCA Spencer, et al. The DNA sequence and biological annotation of human chromosome 1. *Nature*, 441(7091):315–321, 2006.

Thomas L Griffiths, Charles Kemp, and Joshua B Tenenbaum. Bayesian models of cognition. In Ron Sun, editor, *The Cambridge Handbook of Computational Psychology*. Cambridge University Press, 2008.

Roger B Grosse, Chris J Maddison, and Ruslan R Salakhutdinov. Annealing between distributions by averaging moments. *Advances in Neural Information Processing Systems*, pages 2769–2777, 2013.

Roger B Grosse, Zoubin Ghahramani, and Ryan P Adams. Sandwiching the marginal likelihood using bidirectional Monte Carlo. *arXiv preprint arXiv:1511.02543*, 2015.

Yong Gu, Dora E Angelaki, and Gregory C DeAngelis. Neural correlates of multisensory cue integration in macaque MSTd. *Nature Neuroscience*, 11(10):1201–1210, 2008.

Fangjian Guo, Charles Blundell, Hanna Wallach, and Katherine A Heller. The Bayesian echo chamber: Modeling influence in conversations. *arXiv preprint arXiv:1411.2674*, 2014.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83, 1971.

Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.

Geoffrey E Hinton. How neural networks learn from experience. *Scientific American*, 1992.

Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1983.

Daniel R Hochbaum, Yongxin Zhao, Samouil L Farhi, Nathan Klapoetke, Christopher A Werley, Vikrant Kapoor, Peng Zou, Joel M Kralj, Dougal Maclaurin, Niklas Smedemark-Margulies, et al. All-optical electrophysiology in mammalian neurons using engineered microbial rhodopsins. *Nature Methods*, 2014.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500, 1952.

Peter D Hoff. Modeling homophily and stochastic equivalence in symmetric relational data. *Advances in Neural Information Processing Systems*, 20:1–8, 2008.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Douglas N. Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute for Advanced Study, Princeton, 1979.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Patrik O Hoyer and Aapo Hyvarinen. Interpreting neural response variability as Monte Carlo sampling of the posterior. *Advances in neural information processing systems*, pages 293–300, 2003.

Yanping Huang and Rajesh P. N. Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, September 2011.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

Hemant Ishwaran and Mahmoud Zarepour. Exact and approximate sum representations for the Dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283, 2002.

Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. Discovering latent influence in online social activities via shared cascade Poisson processes. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 266–274, 2013.

Mehrdad Jazayeri and Michael N Shadlen. Temporal context calibrates interval timing. *Nature Neuroscience*, 13(8):1020–1026, 2010.

Mehrdad Jazayeri and Michael N Shadlen. A neural mechanism for sensing and reproducing a time interval. *Current Biology*, 25(20):2599–2609, 2015.

Matthew J Johnson. *Bayesian time series models and scalable inference*. PhD thesis, Massachusetts Institute of Technology, June 2014.

Matthew J Johnson and Alan S Willsky. Bayesian nonparametric hidden semi-Markov models. *Journal of Machine Learning Research*, 14(1):673–701, 2013.

Matthew J Johnson and Alan S Willsky. Stochastic variational inference for Bayesian time series models. *Proceedings of the International Conference on Machine Learning*, 32:1854–1862, 2014.

Matthew J Johnson, Scott W Linderman, Sandeep R Datta, and Ryan P Adams. Discovering switching autoregressive dynamics in neural spike train recordings. *Computational and Systems Neuroscience (Cosyne) Abstracts*, 2015.

Lauren M Jones, Alfredo Fontanini, Brian F Sadacca, Paul Miller, and Donald B Katz. Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proceedings of the National Academy of Sciences*, 104(47):18772–18777, 2007.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Eric R Kandel, James H Schwartz, Thomas M Jessell, et al. *Principles of neural science*, volume 4. McGraw-Hill New York, 2000.

David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Network plasticity as Bayesian inference. *PLoS Computational Biology*, 11(11):e1004485, 2015a.

David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Synaptic sampling: A Bayesian approach to neural network plasticity and rewiring. *Advances in Neural Information Processing Systems*, pages 370–378, 2015b.

Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

Jason ND Kerr and Winfried Denk. Imaging in vivo: watching the brain in action. *Nature Reviews Neuroscience*, 9(3):195–205, 2008.

Roozbeh Kiani and Michael N Shadlen. Representation of confidence associated with a decision by neurons in the parietal cortex. *Science*, 324(5928):759–64, May 2009.

John F. C. Kingman. *Poisson Processes (Oxford Studies in Probability)*. Oxford University Press, January 1993. ISBN 0198536933.

David C Knill and Whitman Richards. *Perception as Bayesian inference*. Cambridge University Press, 1996.

Konrad P Körding and Daniel M Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–7, January 2004.

Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic variational inference in Stan. *Advances in Neural Information Processing Systems*, pages 568–576, 2015.

Stephen W Kuffler. Discharge patterns and functional organization of mammalian retina. *Journal of Neurophysiology*, 16(1):37–68, 1953.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Kenneth W Latimer, Jacob L Yates, Miriam LR Meister, Alexander C Huk, and Jonathan W Pillow. Single-trial spike trains in parietal cortex reveal discrete steps during decision-making. *Science*, 349(6244):184–187, 2015.

Tai Sing Lee and David Mumford. Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434–1448, 2003.

Robert Legenstein and Wolfgang Maass. Ensembles of spiking neurons with noise support optimal probabilistic inference in a dynamically changing environment. *PLoS Computational Biology*, 10 (10):e1003859, 2014.

William C Lemon, Stefan R Pulver, Burkhard Höckendorf, Katie McDole, Kristin Branson, Jeremy Freeman, and Philipp J Keller. Whole-central nervous system functional imaging in larval Drosophila. *Nature Communications*, 6, 2015.

Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. The infinite PCFG using hierarchical Dirichlet processes. *Proceedings of Empirical Methods in Natural Language Processing*, pages 688–697, 2007.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

Jeff W Lichtman, Jean Livet, and Joshua R Sanes. A technicolour approach to the connectome. *Nature Reviews Neuroscience*, 9(6):417–422, 2008.

Scott W Linderman and Ryan P. Adams. Discovering latent network structure in point process data. *Proceedings of the International Conference on Machine Learning*, pages 1413–1421, 2014.

Scott W Linderman and Ryan P Adams. Scalable Bayesian inference for excitatory point process networks. *arXiv preprint arXiv:1507.03228*, 2015.

Scott W Linderman and Ryan P Johnson, Matthew Jand Adams. Dependent multinomial models made easy: Stick-breaking with the Pólya-gamma augmentation. *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

Scott W Linderman, Christopher H Stock, and Ryan P Adams. A framework for studying synaptic plasticity with neural spike train data. *Advances in Neural Information Processing Systems*, pages 2330–2338, 2014.

Scott W Linderman, Ryan P Adams, and Jonathan W Pillow. Inferring structured connectivity from spike trains under negative-binomial generalized linear models. *Computational and Systems Neuroscience (Cosyne) Abstracts*, 2015.

Scott W Linderman, Matthew J Johnson, Matthew W Wilson, and Zhe Chen. A nonparametric Bayesian approach to uncovering rat hippocampal population codes during spatial navigation. *Journal of Neuroscience Methods*, 263:36–47, 2016a.

Scott W Linderman, Aaron Tucker, and Matthew J Johnson. Bayesian latent state space models of neural activity. *Computational and Systems Neuroscience (Cosyne) Abstracts*, 2016b.

Fredrik Lindsten, Michael I Jordan, and Thomas B Schön. Ancestor sampling for particle Gibbs. *Advances in Neural Information Processing Systems*, pages 2600–2608, 2012.

Shai Litvak and Shimon Ullman. Cortical circuitry implementing graphical models. *Neural Computation*, 21(11):3010–3056, 2009.

James Robert Lloyd, Peter Orbanz, Zoubin Ghahramani, and Daniel M Roy. Random function priors for exchangeable arrays with applications to graphs and relational data. *Advances in Neural Information Processing Systems*, 2012.

Wei Ji Ma and Mehrdad Jazayeri. Neural coding of uncertainty and probability. *Annual Review of Neuroscience*, 37:205–220, 2014.

Wei Ji Ma, Jeffrey M Beck, Peter E Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–8, November 2006.

David JC MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

Jakob H Macke, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in neural information processing systems*, pages 1350–1358, 2011.

Evan Z Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R Bialas, Nolan Kamitaki, Emily M Martersteck, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, 2015.

Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT Press, 1982.

Paul Miller and Donald B Katz. Stochastic transitions between neural states in taste processing and decision-making. *The Journal of Neuroscience*, 30(7):2559–2570, 2010.

T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023—-1032, 1988.

Shakir Mohamed, Zoubin Ghahramani, and Katherine A Heller. Bayesian and L1 approaches for sparse unsupervised learning. *Proceedings of the International Conference on Machine Learning*, pages 751–758, 2012.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.

Michael L Morgan, Gregory C DeAngelis, and Dora E Angelaki. Multisensory integration in macaque visual cortex depends on cue reliability. *Neuron*, 59(4):662–673, 2008.

Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478, 2008.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, pages 113–162, 2010.

John A Nelder and R Jacob Baker. Generalized linear models. *Encyclopedia of Statistical Sciences*, 1972.

Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, 9(4):e1003037, 2013.

Mark EJ Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics (SIAM) Review*, 45(2):167–256, 2003.

Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

Seung Wook Oh, Julie A Harris, Lydia Ng, Brent Winslow, Nicholas Cain, Stefan Mihalas, Quanxin Wang, Chris Lau, Leonard Kuan, Alex M Henry, et al. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*, volume 3. Clarendon Press, 1978.

Peter Orbanz and Daniel M Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):437–461, 2015.

Peter Orbanz and Yee Whye Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. Springer, 2011.

Adam M Packer, Darcy S Peterka, Jan J Hirtz, Rohit Prakash, Karl Deisseroth, and Rafael Yuste. Two-photon optogenetics of dendritic spines and neural circuits. *Nature Methods*, 9(12):1202–1205, 2012.

Liam Paninski. Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15(4):243–262, January 2004.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29(1-2):107–126, 2010.

Andrew V Papachristos. Murder by structure: Dominance relations and the social structure of gang homicide. *American Journal of Sociology*, 115(1):74–128, 2009.

Il Memming Park and Jonathan W Pillow. Bayesian spike-triggered covariance analysis. *Advances in Neural Information Processing Systems*, pages 1692–1700, 2011.

Patrick O Perry and Patrick J Wolfe. Point process modelling for directed interaction networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2013.

Biljana Petreska, Byron Yu, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Dynamical segmentation of single trials from population neural data. *Advances in Neural Information Processing Systems*, pages 756–764, 2011.

David Pfau, Eftychios A Pnevmatikakis, and Liam Paninski. Robust learning of low-dimensional dynamics from large neural ensembles. *Advances in Neural Information Processing Systems*, pages 2391–2399, 2013.

Jonathan W. Pillow and James Scott. Fully Bayesian inference for neural models with negative-binomial spiking. *Advances in Neural Information Processing Systems*, pages 1898–1906, 2012.

Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

Eftychios A Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, et al. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 2016.

Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using Pólya-gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349, 2013.

Ruben Portugues, Claudia E Feierstein, Florian Engert, and Michael B Orger. Whole-brain activity maps reveal stereotyped, distributed networks for visuomotor behavior. *Neuron*, 81(6):1328–1343, 2014.

Alexandre Pouget, Jeffrey M Beck, Wei Ji Ma, and Peter E Latham. Probabilistic brains: knowns and unknowns. *Nature Neuroscience*, 16(9):1170–1178, 2013.

Robert Prevedel, Young-Gyu Yoon, Maximilian Hoffmann, Nikita Pak, Gordon Wetzstein, Saul Kato, Tina Schrödel, Ramesh Raskar, Manuel Zimmer, Edward S Boyden, et al. Simultaneous whole-animal 3d imaging of neuronal activity using light-field microscopy. *Nature Methods*, 11(7): 727–730, 2014.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Adrian E Raftery and Steven Lewis. How many iterations in the Gibbs sampler? *Bayesian Statistics*, pages 763–773, 1992.

Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 33:275––283, 2014.

Rajesh P. N. Rao. Bayesian computation in recurrent neural circuits. *Neural Computation*, 16(1): 1–38, January 2004.

Rajesh P. N. Rao. Neural models of Bayesian belief propagation. In *Bayesian brain: Probabilistic approaches to neural computation*, pages 236–264. MIT Press Cambridge, MA, 2007.

Rajesh P. N. Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, January 1999.

Danilo J Rezende, Daan Wierstra, and Wulfram Gerstner. Variational learning for recurrent spiking networks. *Advances in Neural Information Processing Systems*, pages 136–144, 2011.

Fred Rieke, David Warland, Rob de Ruyter van Steveninck, and William Bialek. *Spikes: exploring the neural code*. MIT press, 1999.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

Maneesh Sahani. *Latent variable models for neural data analysis*. PhD thesis, California Institute of Technology, 1999.

Maneesh Sahani and Peter Dayan. Doubly distributional population codes: simultaneous representation of uncertainty and multiplicity. *Neural Computation*, 2279:2255–2279, 2003.

Joshua R Sanes and Richard H Masland. The types of retinal ganglion cells: current status and implications for neuronal classification. *Annual Review of Neuroscience*, 38:221–246, 2015.

Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

Ben Shababo, Brooks Paige, Ari Pakman, and Liam Paninski. Bayesian inference and online experimental design for mapping neural microcircuits. *Advances in Neural Information Processing Systems*, pages 1304–1312, 2013.

Vahid Shalchyan and Dario Farina. A non-parametric Bayesian approach for clustering and tracking non-stationarities of neural spikes. *Journal of Neuroscience Methods*, 223:85–91, 2014.

Lei Shi and Thomas L Griffiths. Neural implementation of hierarchical Bayesian inference by importance sampling. *Advances in Neural Information Processing Systems*, 2009.

Yousheng Shu, Andrea Hasenstaub, and David A McCormick. Turning on and off recurrent balanced cortical activity. *Nature*, 423(6937):288–293, 2003.

Jack W Silverstein. The spectral radii and norms of large dimensional non-central random matrices. *Stochastic Models*, 10(3):525–532, 1994.

Aleksandr Simma and Michael I Jordan. Modeling events with cascades of Poisson processes. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2010.

Eero P Simoncelli. Optimal estimation in sensory systems. *The Cognitive Neurosciences, IV*, 2009.

Anne C Smith and Emery N Brown. Estimating a state-space model from point process observations. *Neural Computation*, 15(5):965–91, May 2003.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

Sen Song, Kenneth D Miller, and Lawerence F Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticitye. *Nature Neuroscience*, 3(9):919–26, September 2000. ISSN 1097-6256.

Daniel Soudry, Suraj Keshri, Patrick Stinson, Min-hwan Oh, Garud Iyengar, and Liam Paninski. Efficient "shotgun" inference of neural connectivity from highly sub-sampled activity data. *PLoS Computational Biology*, 11(10):1–30, 10 2015. doi: 10.1371/journal.pcbi.1004464.

Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS Computational Biology*, 1(4):e42, 2005.

Olav Stetter, Demian Battaglia, Jordi Soriano, and Theo Geisel. Model-free reconstruction of excitatory neuronal connectivity from calcium imaging signals. *PLoS Computational Biology*, 8(8): e1002653, 2012.

Ian Stevenson and Konrad Koerding. Inferring spike-timing-dependent plasticity from spike train data. *Advances in Neural Information Processing Systems*, pages 2582–2590, 2011.

Ian H Stevenson, James M Rebesco, Nicholas G Hatsopoulos, Zach Haga, Lee E Miller, and Konrad P Körding. Bayesian inference of functional connectivity and network structure from spikes. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(3):203–213, 2009.

Alan A Stocker and Eero P Simoncelli. Noise characteristics and prior expectations in human visual speed perception. *Nature Neuroscience*, 9(4):578–85, April 2006.

Yee Whye Teh and Michael I Jordan. Hierarchical Bayesian nonparametric models with applications. *Bayesian Nonparametrics*, pages 158–207, 2010.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006.

Joshua B Tenenbaum, Thomas L Griffiths, and Charles Kemp. Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318, 2006.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.

Luke Tierney and Joseph B Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

Wilson Truccolo, Uri T. Eden, Matthew R. Fellows, John P. Donoghue, and Emery N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005. doi: 10.1152/jn.00697.2004.

Philip Tully, Matthias Hennig, and Anders Lansner. Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Frontiers in Synaptic Neuroscience*, 6(8), 2014.

Srini Turaga, Lars Buesing, Adam M Packer, Henry Dalgleish, Noah Pettit, Michael Hausser, and Jakob Macke. Inferring neural population dynamics from multiple partial recordings of the same neural circuit. *Advances in Neural Information Processing Systems*, pages 539–547, 2013.

Leslie G Valiant. *Circuits of the Mind*. Oxford University Press, Inc., 1994.

Leslie G Valiant. Memorization and association on a realistic neural model. *Neural Computation*, 17(3):527–555, 2005.

Leslie G Valiant. A quantitative theory of neural computation. *Biological Cybernetics*, 95(3):205–211, 2006.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. Beam sampling for the infinite hidden Markov model. *Proceedings of the International Conference on Machine Learning*, pages 1088–1095, 2008.

Michael Vidne, Yashar Ahmadian, Jonathon Shlens, Jonathan W Pillow, Jayant Kulkarni, Alan M Litke, EJ Chichilnisky, Eero Simoncelli, and Liam Paninski. Modeling the impact of common noise inputs on the network activity of retinal ganglion cells. *Journal of Computational Neuroscience*, 33 (1):97–121, 2012.

Joshua T Vogelstein, Brendon O Watson, Adam M Packer, Rafael Yuste, Bruno Jedynak, and Liam Paninski. Spike inference from calcium imaging using sequential Monte Carlo methods. *Biophysical Journal*, 97(2):636–655, 2009.

Joshua T Vogelstein, Adam M Packer, Timothy A Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of Neurophysiology*, 104(6):3691–3704, 2010.

Hermann von Helmholtz and James Powell Cocke Southall. *Treatise on Physiological Optics: Translated from the 3rd German Ed.* Optical Society of America, 1925.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Yair Weiss, Eero P Simoncelli, and Edward H Adelson. Motion illusions as optimal percepts. *Nature Neuroscience*, 5(6):598–604, 2002.

Mike West, P Jeff Harrison, and Helio S Migon. Dynamic generalized linear models and Bayesian forecasting. *Journal of the American Statistical Association*, 80(389):73–83, 1985.

John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans: the mind of a worm. *Philosophical Transactions of the Royal Society of London: Series B (Biological Sciences)*, 314:1–340, 1986.

Louise Whiteley and Maneesh Sahani. Attention in a Bayesian framework. *Frontiers in Human Neuroscience*, 6, 2012.

Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping subsecond structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

Jesse Windle, Nicholas G Polson, and James G Scott. Sampling Pólya-gamma random variates: alternate and approximate techniques. *arXiv preprint arXiv:1405.0506*, 2014.

Frank Wood and Michael J Black. A nonparametric Bayesian alternative to spike sorting. *Journal of Neuroscience Methods*, 173(1):1–12, 2008.

Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. *arXiv preprint arXiv:1507.00996*, 2015.

Tianming Yang and Michael N Shadlen. Probabilistic reasoning by neurons. *Nature*, 447(7148): 1075–80, June 2007.

Byron M. Yu, John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, 102:614–635, 2009.

Alan Yuille and Daniel Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.

Richard S Zemel, Peter Dayan, and Alexandre Pouget. Probabilistic interpretation of population codes. *Neural Computation*, 10(2):403–30, February 1998.

Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 16, 2013.

Mingyuan Zhou, Lingbo Li, Lawrence Carin, and David B Dunson. Lognormal and gamma mixed negative binomial regression. *Proceedings of the International Conference on Machine Learning*, pages 1343–1350, 2012.